ウェブアプリについて
PMが知っておくべきこと

基礎編

凡例

- フリーソフトウェア（青）
- PM演習で使うもの（下線）
- ［括弧の中は憶えなくてよい］

アプリの選択肢

- ネイティブアプリケーション
  - Windows
  - Mac OS
  - Android
  - iOS (iPhone, iPad)
- ウェブアプリケーション
  - HTML5 (HTML + CSS + JavaScript)
  - Flash：iOSは不可（他も実行環境が必要）
  - Javaアプレット：同上

ウェブアプリケーション

- メリット
  - ブラウザがあれば動く。
- デメリット
  - ブラウザごとに動作が異なる。
    - 違いを吸収するようなライブラリを使えばよい。
  - ローカルファイルなどの利用は制限される。
    - 将来はすべてクラウドに置くようになる。
  - 端末の性能をすべて引き出せるわけではない。

ウェブアプリの基本構成

http://...の「http」はプロトコルの指示
httpsは暗号通信（SSL）

クライアント側

サーバ側

クライアント側

- OS：なんでもよい
- ブラウザ
  - Internet Explorer (IE)
  - Chrome, Chromium
  - Firefox
  - Safari・Opera（エンジンが同じになる予定）

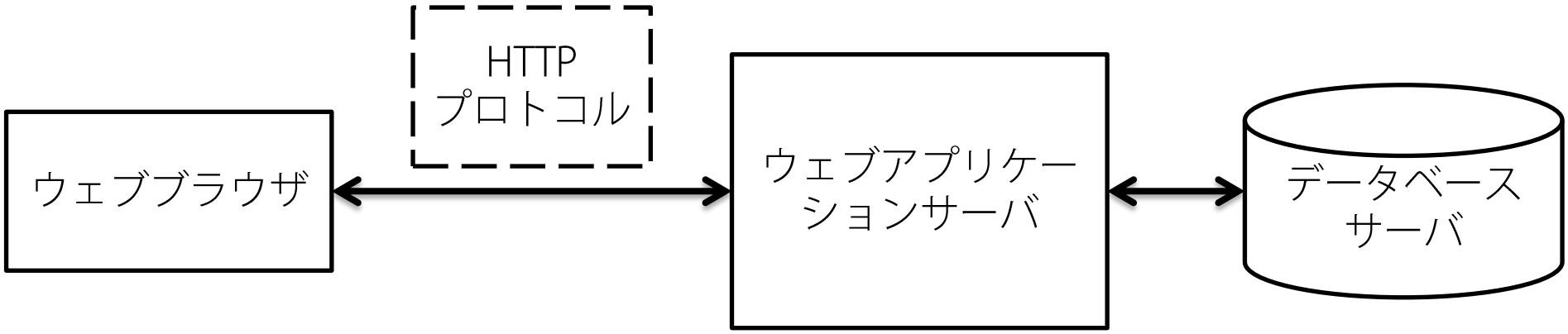
ブラウザのための
プログラミング言語

- JavaScript（デファクト）
- ［Flash (ActionScript)］
- ［Java（アプレット）］

サーバ側

- OS
  - Windows
    - Windows 7, 8は開発専用。運用にはWindows Server。
  - Unix系
    - Linux
    - ［BSD系。FreeBSD・OpenBSDなど］
    - ［ Mac OS X］
    - ［ Oracle Solaris］
    - ［ IBM AIX］
    - …

ウェブアプリの基本構成2



クライアント側　　サーバ側

ウェブアプリケーションサーバ

- IIS（Internet Information Service）
  - Windows上でのみ動く。
- Apache HTTP Server （略してApache）
  - Unix系OSで使う。Windows上でも一応動く
- ［nginx］（速いという噂）
  - Unix系OSで使う。

ウェブアプリケーションサーバのためのプログラミング言語

- Java
- P (Perl, PHP, Python)
- Ruby
- C#（IIS上のみ）
- JavaScript（node.js）これが普及すると、クライアント側とサーバ側が同じ言語になる！

データベースサーバ

- リレーショナルデータベース
  - MySQL
  - PostgreSQL
  - SQLite（組み込みで使われている）
  - Oracle
  - DB2 (IBM)
  - SQL Server (MS)：Windows上でのみ動く。
- リレーショナルデータベース以外のものも流行ってはいる。

開発に必要なもの

- Apache
- PHP
- MySQL

} Linuxならコマンド1行で準備完了
WindowsではXAMPPを使うのが簡単

- テキストエディタ
- ブラウザ

テキストエディタの代わりに、Eclipseや
Aptana、NetBeans、Dreamweaver、Visual
Studioを使ってもよい。

ここからが本番です。

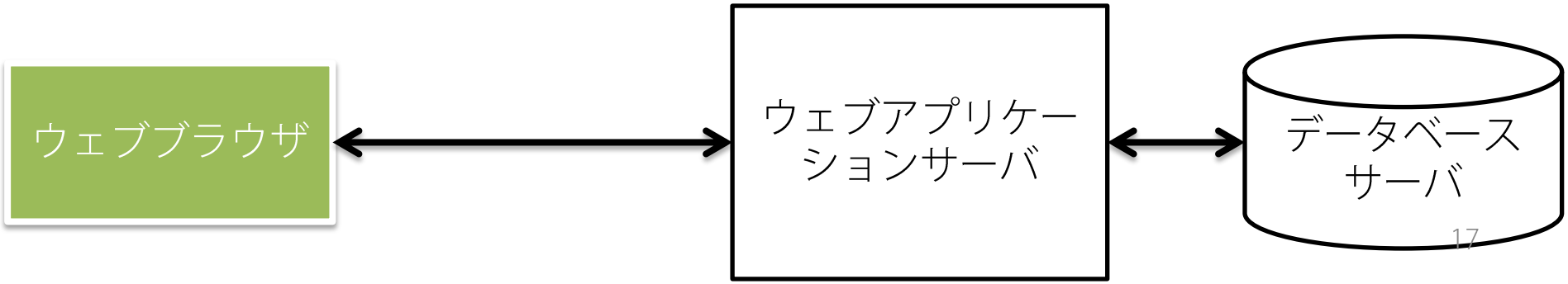
動かしてみる経験が大事。

クライアント側

HTML
CSS
JavaScript

HTML
Hyper Text Markup Language

ウェブページの構造の指定



HTML

1. Document Rootにファイルを置く。
   XAMPPの場合は
   c:/xampp/htdocs/
2. http://localhost/hello.html にアクセスする。
   localhostは自分のマシンのこと

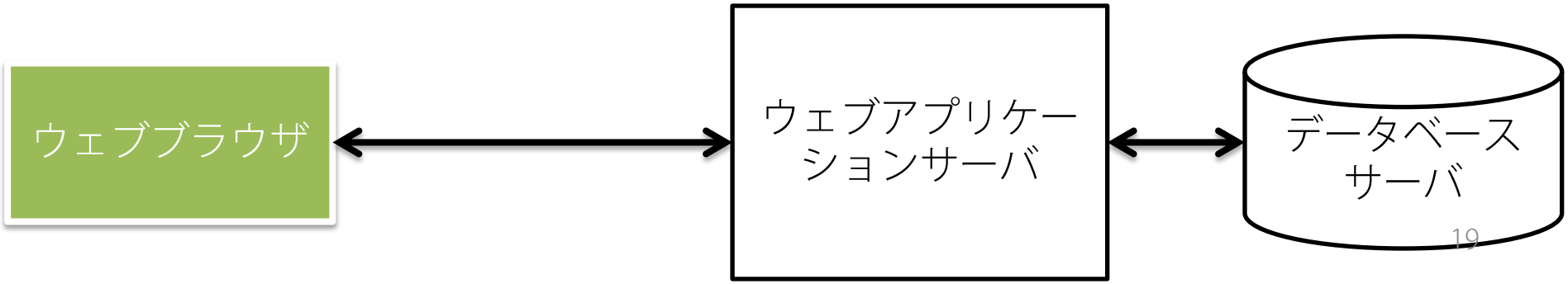
hello.html

```
<!DOCTYPE html>
<html>
<head>
<title>HELLO</title>
</head>
<body>
<p>Hello, World!</p>
</body>
</html>
```

CSS
Cascading Style Sheet

ウェブページの「見た目」の指定

例：http://www.csszengarden.com/



CSS

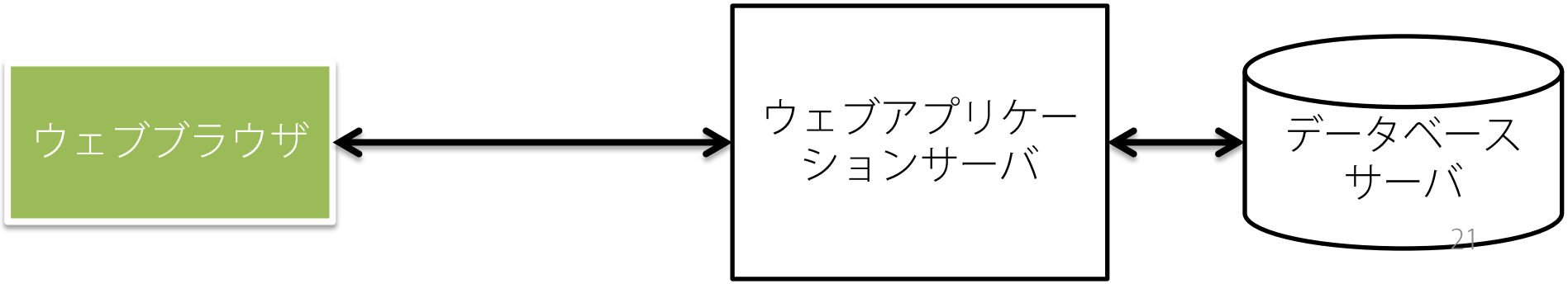
1. hello.htmlと同じ場所にCSSファイルを置く。
2. hello.htmlのhead要素内に以下を追記する。
   `<link rel="stylesheet" href="style.css" />`
3. http://localhost/hello.html にアクセスする。

style.css

```
body {
  color: white;
  background-color: black;
}
```

JavaScript

HTMLは静的。動的にしたいならプログラムが必要。
ブラウザ動くプログラムはJavaScript。



JavaScript

1. hello.htmlと同じ場所にJSファイルを置く。
2. hello.htmlのbody要素の最後に以下を追記する。

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

3. http://localhost/hello.html にアクセスする。

script.js

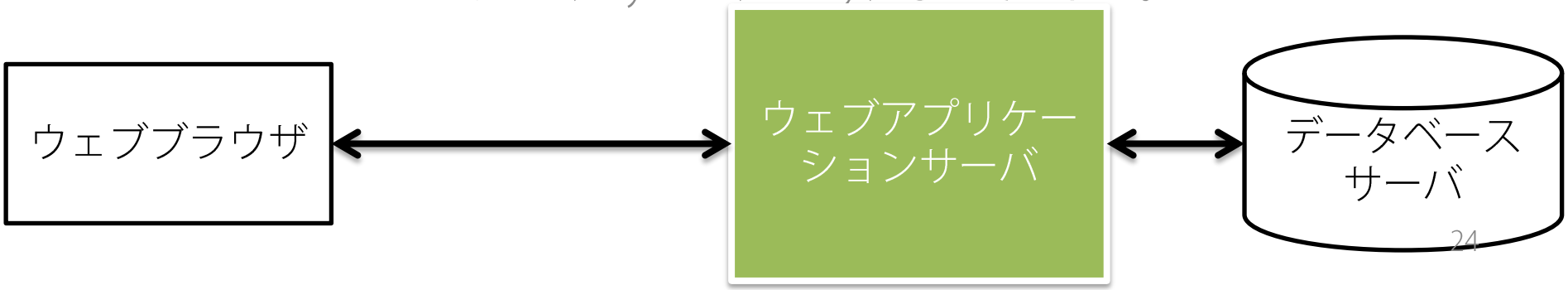
```
document.write(new Date());
```

サーバ側

PHP
SQL

PHP

サーバ側で動くプログラム。
「なぜPHP？」
よく使われていて資料も多い。
JavaやPerl、Python、Ruby、なんでもよい。



PHP

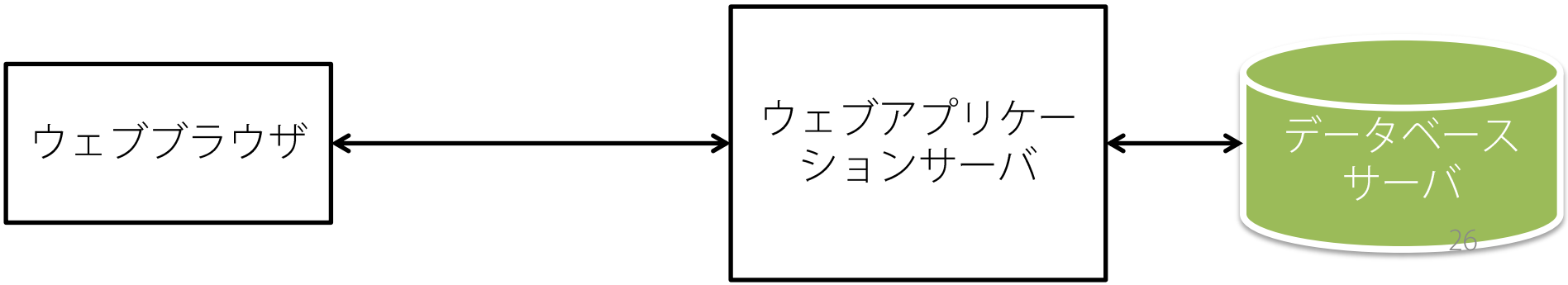
1. hello.htmlと同じ場所にPHPファイルを置く。
2. http://localhost/hello.php?hoge=Yabuki にアクセスする。URL（URI）を使っていろんなリクエストができる。（設計が必要）

hello.php
（セキュリティホールあり）

```
<!DOCTYPE html>
<html>
<head>
<title>HELLO</title>
</head>
<body>
<?php
  echo 'Hello, ' . $_GET['hoge'];
?>
</body>
</html>
```

SQL

リレーショナルデータベースのための唯一の言語。
欲しい結果を書くだけだから、慣れると便利。

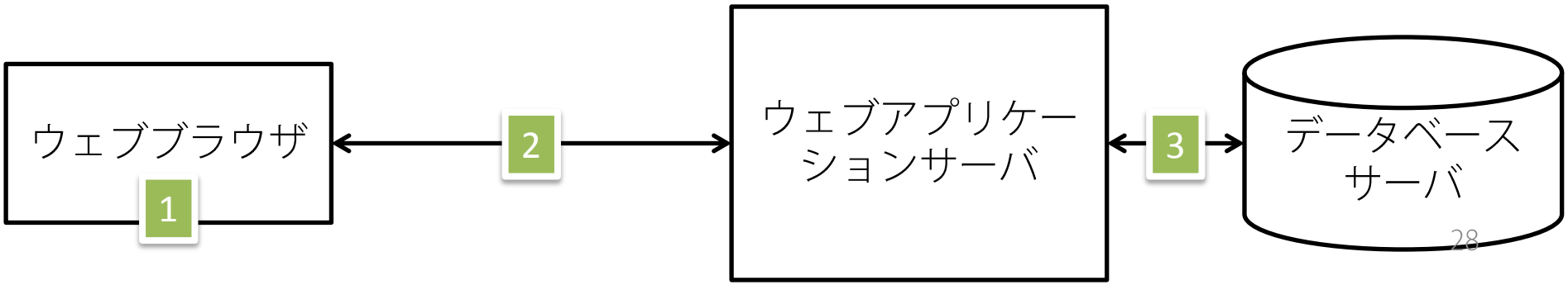


SQL

1. データベースの作成
```
CREATE DATABASE mydb
CHARSET=utf8;
```
2. テーブルの作成
```
USE mydb;
CREATE TABLE people (
  id INT PRIMARY KEY,
  name VARCHAR(20)
);
```
3. データの生成（Create）
```
INSERT INTO people
(id,name) VALUES
(1,'Alice'),(2,'Bob');
```
4. データの読み込み（Read）
```
SELECT name FROM
people;
```
5. データの更新（Update）
```
UPDATE people SET
name='Bobby' WHERE
id=2;
```
6. データの削除（Delete）
```
DELETE FROM people
WHERE id=1;
```

やり残していること

1. JavaScriptでHTMLを操作する方法（DOMの理解）
2. JavaScriptでサーバ（PHP）とデータをやりとりする方法（Ajax）
3. PHPでMySQLを利用する方法



できるようになるまで練習

- http://localhost/test/hello.html で「Hello, World!」と表示されるページを作る。
- hello.htmlにアクセスすると、その日時が表示されるようにする（JavaScriptで）
- http://localhost/test/hello.php?foo=Bar で、「Hello, Bar!」と表示されるページを作る。
- 適当なテーブルを作り、データのCRUDのためのSQL文を書く。

応用編

これまで紹介してきたことを積み重ねてウェブアプリを作る

でも、ちょっと面倒。

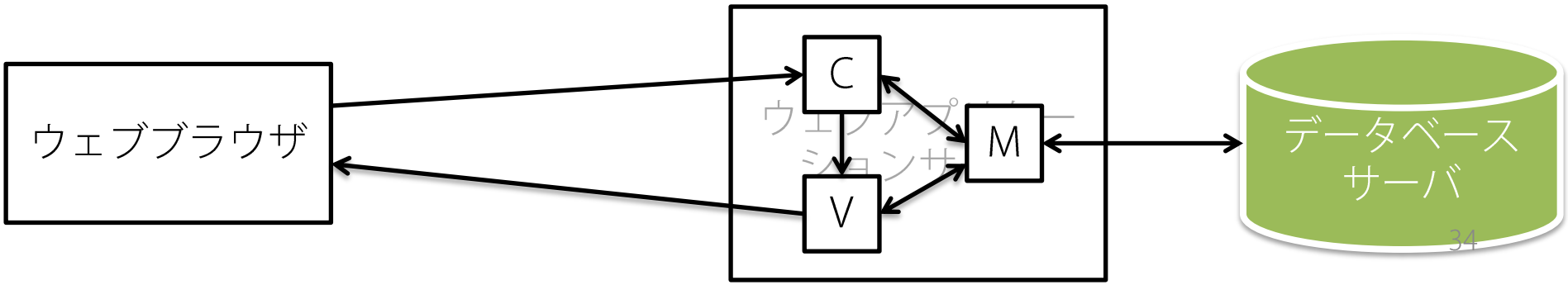
フレームワークの導入

ウェブアプリは
似たパターンが多いから。

ウェブアプリケーションフレームワーク

- Ruby
  - Ruby on Rails（これがすべてを変えた）
- PHP
  - CakePHP
  - Symfony
  - …

Model View Controller（MVC）パターン



convention over configuration, CoC 設定より規約：CakePHPの場合1

1. 複数形の名前のテーブルを作る。(**c**ustomer**s**)
2. リクエストに対応するためのコントローラを作る（キャメルケース）。(Controller/**C**ustomer**s****C**ontroller.php) http://localhost/.../**c**ustomer**s**/ でアクセス。
3. データを処理するモデルを作る。(Model/**C**ustomer.php)
4. 表示のためのビューを作る。(View/**C**ustomer**s**/index.ctpなど)

設定より規約：CakePHPの場合2
テーブルの要素

- 名前は単数形。
- 主キーの名前は「id」。
- 外部キー（関連するテーブルのid）は「テーブル名_id」に格納する。
- 「name」には、表示に使う名前を格納する。
- 「created」には、データが生成された日時が自動的に格納される。
- 「modified」には、データが更新された日時が自動的に格納される。

規約に従って準備をすれば、大部分を自動化できる。

CakePHPの仕様に詳しくなるのは今ではない。

PM演習についてのまとめ

- ウェブアプリケーション
- 開発言語
  - PHP（サーバ側）
  - JavaScript（クライアント側）
  - SQL（データベースサーバ）
- ウェブサーバはApache HTTP Server
- データベースサーバはMySQL
- フレームワークはCakePHP
- （バージョン管理システムはGit）
- （バグトラッキングはGitHub上）