

# LARAVEL API

คู่มือการใช้งาน **Laravel** เพื่อทำ **Api** ในรูปแบบต่างๆ

# แนะนำตัว

## นายศรัณย์ สุทธิภู (ตั้ม)

ทำงานที่ เนชั่นมัลติมีเดียกรุ๊ป

ตำแหน่ง Backend Developer

เบอร์ติดต่อ 090-913-1190

## ประสบการณ์ทำงาน

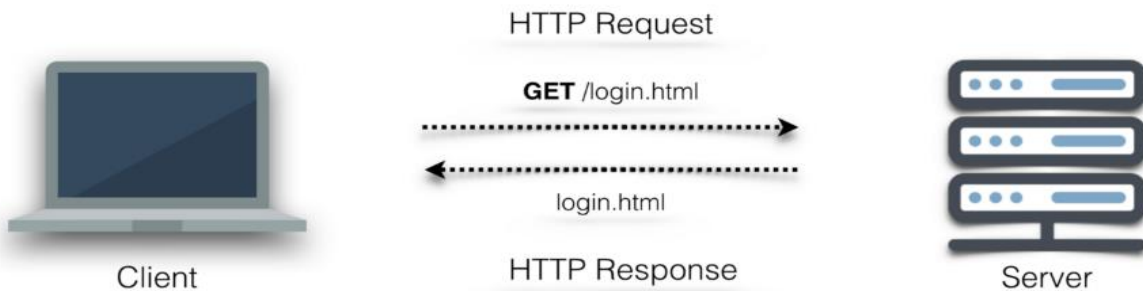
- เขียน api ให้แอปพลิเคชัน 3 ปี
- พัฒนาระบบออกใบรับรองให้กับกรมวิชาการเกษตร
- ทำเว็บสำหรับสอบทดสอบผู้สมัครงานครู
- ทำระบบรับลงทะเบียนให้กับโรงเรียนสมุทรปราการ

# การทำงานของเราภาษาต่าง ๆ

คู่มือการใช้งาน **Laravel** เพื่อทำ **Api** ในรูปแบบต่างๆ

# การทำงานของ web browser

เมื่อผู้ใช้กรอก URL (Uniform Resource Locator) หรือ Web Address เช่น Facebook.com, Youtube.com หรือ Twitter.com ลงไป มันจะรับข้อมูลของเว็บไซต์นั้นๆ ที่เขียนขึ้นด้วยภาษา HTML (HyperText Markup Language) และส่งผ่านมาจาก Hypertext Transfer Protocol มาประมวลผลโค้ดต่างๆ จากนั้นทำการแสดงผลข้อมูลในเว็บไซตนั้นๆ ขึ้นบนอุปกรณ์ของผู้ใช้



# PHP คืออะไร

PHP – Personal Home Page Tool (อ่านว่า พีเอชพี) และ PHP คืออะไร จริงๆแล้วเป็น **computer language** ในรูปแบบ **Server-side scripting** ใช้ในการสร้าง **web page** เป็น **open source** โดยแสดงในรูปแบบ **HTML** มีวัตถุประสงค์เพื่อใช้ในการ พัฒนา **web site** และ **web page** ความสามารถของ **PHP** นั้น สามารถที่จะทำงานเกี่ยวกับ **Dynamic Web** ได้ทุกรูปแบบ เหมือนกับ **CGI** หรือ **ASP** ไม่ว่าจะเป็นการดูแลจัดการระบบฐานข้อมูล ระบบรักษาความปลอดภัยของเว็บเพจ การรับ – ส่ง **Cookies** เป็นต้น



# MySQL คืออะไร

**MySQL** เป็นระบบจัดการฐานข้อมูลเชิงสัมพันธ์โอเพนซอร์สบนพื้นฐานของ SQL ซึ่ง **MySQL** ได้รับการออกแบบและปรับให้เหมาะสมสำหรับเว็บแอปพลิเคชันและสามารถทำงานบนแพลตฟอร์มใดก็ได้ **MySQL** ทำงานเป็นดาต้าเบสเซิร์ฟเวอร์และอนุญาตให้ผู้ใช้หลายคนจัดการและสร้างฐานข้อมูลจำนวนมาก มันเป็นองค์ประกอบสำคัญใน



# HTML คืออะไร

ภาษาหลักที่ใช้ในการเขียนเว็บเพจ โดยใช้ **Tag** ในการกำหนดการแสดงผล **HTML** ย่อมาจากคำว่า **Hypertext Markup Language** โดย **Hypertext** หมายถึงข้อความที่เชื่อมต่อกันผ่านลิงค์(**Hyperlink**) **Markup language** หมายถึงภาษาที่ใช้ **Tag** ในการกำหนดการแสดงผลสิ่งต่างๆที่แสดงอยู่บนเว็บเพจ ดังนั้น **HTML** จึงหมายถึงภาษาที่ใช้ **Tag** ในการกำหนดการแสดงผลเว็บเพจที่ต่างก็เชื่อมถึงกันใน **Hyperspace** ผ่าน **Hyperlink** นั้นเองปัจจุบันมีการพัฒนาและกำหนดมาตรฐานโดยองค์กร **World Wide Web Consortium (W3C)**

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="UTF-8">
5      <title>Title goes here</title>
6    </head>
7    <body>
8
9    </body>
10 </html>
```

# CSS คืออะไร

ภาษา **CSS** หรือที่ชื่อเต็มๆ คือ "Cascading Style Sheets" มันเป็นภาษาที่ใช้พัฒนาลักษณะรูปแบบ ใส่พื้นหลัง หรือเพิ่มกรอบข้อความ ของหน้าเว็บ เพื่อเพิ่มความสวยงามให้หน้าเว็บของเรา **CSS** สามารถกำหนดรูปแบบพร้อมกันทีเดียวได้ ทำให้เวลาแก้ไขไม่ต้องคอยแก้ทีละส่วน



**Tailwind CSS**

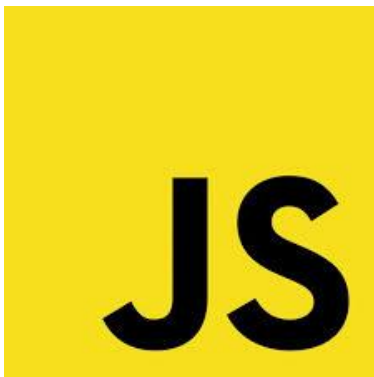


```
body {  
  font: x-small;  
  background: #  
  color: black;  
  margin: 0;  
  padding: 0;
```



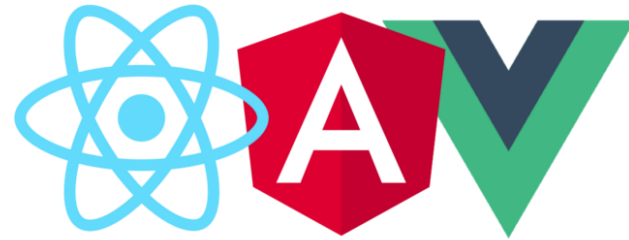
# JAVASCRIPT คืออะไร

ภาษา **JavaScript** หรือย่อ **JS** เป็นภาษาเขียนโปรแกรมที่ถูกพัฒนาและปฏิบัติตามข้อกำหนดมาตรฐานของ **ECMAScript**; ภาษา **JavaScript** นั้นเป็นภาษาระดับสูง ถือว่าเป็นเทคโนโลยีหลักของการพัฒนาเว็บไซต์ (World Wide Web) มันทำให้หน้าเว็บสามารถตอบโต้กับผู้ใช้ได้โดยที่ไม่จำเป็นต้องรีเฟรชหน้าใหม่ (Dynamic website) เว็บไซต์จำนวนมากใช้ภาษา **JavaScript** สำหรับควบคุมการทำงานที่ฝั่ง **Client-side** นั้นทำให้เว็บเบราว์เซอร์ต่างๆ มี **JavaScript engine** ที่ใช้สำหรับประมวลผลสคริปของภาษา **JavaScript** ที่รันบนเว็บเบราว์เซอร์



# FRAMEWORK คืออะไร

อาจหมายถึง ชุดคำสั่ง เครื่องมือ หรือโครงสร้างอย่างใดอย่างหนึ่ง  
ที่สร้างขึ้นมาเพื่ออำนวยความสะดวกแก่ผู้ใช้งาน ซึ่ง **Framework**  
มีหลายประเภท หลายแบบ และจะมีวิธีการใช้ที่คล้ายๆกัน



# XAMLL คืออะไร

**Xampp** คือโปรแกรมสำหรับจำลองเครื่องคอมพิวเตอร์ส่วนบุคคลของเราให้ทำงานในลักษณะของ **Webserver** ซึ่งเครื่องคอมพิวเตอร์ของเราจะเป็นทั้งเครื่องแม่ และเครื่องลูกในเครื่องเดียวกัน โดยไม่ต้องเชื่อมต่อกับ **Internet** ก็สามารถทดสอบกับเว็บไซต์ที่เราสร้างขึ้นมาได้ทุกที่ทุกเวลา อีกทั้งยังประหยัดเวลาและไม่มีค่าใช้จ่ายใดๆ ทั้งสิ้น ซึ่งในปัจจุบันนี้ได้รับความนิยมจากผู้ใช้ **CMS** ในการสร้างเว็บไซต์

**XAMPP** ประกอบด้วย **Apache, PHP, MySQL, PHP MyAdmin, Perl** ซึ่งเป็นโปรแกรมพื้นฐานที่รองรับการทำงาน **CMS** เป็นชุดโปรแกรม สำหรับออกแบบเว็บไซต์ที่ได้รับความนิยมในปัจจุบัน ไฟล์สำหรับติดตั้ง **XAMPP** นั้นอาจมีขนาดใหญ่สักหน่อย เนื่องจาก มีชุดควบคุมการทำงานที่ช่วยให้การปรับแต่งส่วนต่าง ๆ ง่ายขึ้น **XAMPP** นั้นรองรับระบบปฏิบัติการหลายตัว เช่น **Windows, Linux, MacOS**



# XAMPP

# ทบทวนการเขียน PHP เบื้องต้น

คู่มือการใช้งาน **Laravel** เพื่อทำ **Api** ในรูปแบบต่างๆ

# การประกาศตัวแปร

ตัวแปรเป็นสัญลักษณ์แทนสิ่งใดสิ่งหนึ่งที่เราต้องการ เพื่อนำไปใช้ประมวลผล เช่น `$a=1`; โดยเราสามารถกำหนดตัวแปรไว้ล่วงหน้าแล้วเรียกใช้งานได้ตลอดหน้า **PHP** นั้น

## กฎการตั้งชื่อตัวแปร

- ❑ เริ่มต้นด้วย `$`
- ❑ ตามด้วย **A-Z** หรือ **a-z** หรือ **0-9** หรือ `_` เช่น `$myvar`; `$my_var`; `$myVar`;
- ❑ **Case Sensitive** ตัวพิมพ์ใหญ่/เล็กถือเป็นคนละตัว เช่น `$myvar`; `$Myvar`; `$MyVar`; `$myVar`;
- ❑ ไม่ตั้งชื่อซ้ำค่าสงวน เช่น `$_POST`; `$_SESSION`; `$_GET`;

# ชนิดตัวแปร

- ❑ Boolean -> True , False
- ❑ Integer -> เลขจำนวนเต็ม
- ❑ Float -> เลขจำนวนจริง
- ❑ String -> ตัวอักษรที่นำไปคำนวณทางคณิตศาสตร์ไม่ได้
- ❑ Array -> ตัวแปรชุด
- ❑ Object -> เก็บคุณสมบัติของ Object
- ❑ Resource -> สำหรับอ้างอิงถึงแหล่งภายนอก เช่น การเปิดไฟล์ข้อมูล การเชื่อมต่อฐานข้อมูล
- ❑ Null -> ตัวแปรที่ไม่มีค่าอะไรเลยเรียกว่ามีค่าเป็น Null เช่น เมื่อประกาศตัวแปรแล้วแต่ยังไม่ได้กำหนดค่าใดๆให้ตัวแปร กำหนดค่าให้ตัวแปรมีค่าเป็น Null `$MySalary = NULL;`

```
$MyName = 'Manop Kongoon'; //String Variable
$RoomNo='405'; //String Variable
$maximum_score = 100; //Integer Variable
$is_a_student = true; //Boolean Variable
$TotalScore=10+30;
$Score1=10;
$Score2=30;
$TotalScore=$Score1+$Score2;
```

# คำสั่งเงื่อนไข if, if else

ในการเขียนโปรแกรม อาจจะมีเงื่อนไขหรือข้อกำหนดบางอย่างที่คุณต้องการให้โปรแกรมทำงานแตกต่างกันไป การตัดสินใจจึงเป็นเรื่องธรรมดาที่เกิดขึ้นทั้งในการเขียนโปรแกรมและในชีวิตประจำวัน เงื่อนไขการตัดสินใจคือ  $>$ ,  $<$ ,  $>=$ ,  $<=$ ,  $=$ ,  $!=$

```
If (เงื่อนไข) {  
    จริง  
}
```

```
If (เงื่อนไข) {  
    จริง  
} else {  
    เท็จ  
}
```

```
If (เงื่อนไข 1) {  
    จริง  
} If (เงื่อนไข 2) {  
} else {  
    เท็จ  
}
```

# คำสั่งวนซ้ำ for loop

คำสั่ง **For loop** คือคำสั่งวนซ้ำที่มีการทำงานในจำนวนรอบที่แน่นอน ซึ่งสามารถกำหนดค่าเริ่มต้น เงื่อนไข และการเปลี่ยนแปลงไว้ที่เดียวกันในตอนต้นของ **Loop** ทำให้การเขียนโปรแกรมสั้นและกระชับมากขึ้น นี่เป็นรูปแบบของการใช้งานคำสั่ง **For loop** ในภาษา **PHP**

```
for ($i = 1; $i <= 10; $i++) {  
    echo "$i, ";  
}  
echo "\n";
```

ผลลัพธ์

```
1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
```

```
for ($i = 0; $i <= 50; $i += 5) {  
    echo "$i, ";  
}  
echo "\n";
```

ผลลัพธ์

```
0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50,
```



# คำสั่งวนซ้ำ foreach loop

คำสั่ง **Foreach loop** ถูกออกแบบมาให้ใช้งานกับอาเรย์ มันใช้สำหรับวนอ่านค่าในอาเรย์โดยจะเริ่มจากสมาชิกตัวแรกจนถึงสมาชิกตัวสุดท้าย มาดูตัวอย่างการใช้งาน **Foreach** ในภาษา **PHP**

```
// foreach with normal array
$numbers = [10, 20, 30, 40, 50];
foreach ($numbers as $el) {
    echo "$el\n";
}
```

```
10
20
30
40
50
```

```
// foreach with key value array
$country["us"] = "United State";
$country["de"] = "German";
$country["uk"] = "Ukraine";
$country["sk"] = "Slovakia";
foreach ($country as $key => $value) {
    echo "$key = $value\n";
}
```

```
us = United State
de = German
uk = Ukraine
sk = Slovakia
```

# การสร้าง Function

การสร้างฟังก์ชันเป็นการรวบรวมกลุ่มคำสั่งของโปรแกรมในการทำงานที่เฉพาะเจาะจง การสร้างฟังก์ชันจะทำให้คุณสามารถเรียกใช้โค้ดเดิม (**reuse**) โดยที่ไม่ต้องเขียนขึ้นใหม่ มาดูตัวอย่างการสร้างฟังก์ชันในภาษา PHP

```
function welcome() {  
    echo "Welcome to marcuscode.\n";  
}  
  
function hello($name) {  
    echo "Hello $name!\n";  
}
```

```
function get_site_name() {  
    return "marcuscode";  
}  
  
function factorial($n) {  
    $fac = 1;  
    for ($i = 1; $i <= $n; $i++) {  
        $fac *= $i;  
    }  
    return $fac;  
}
```

# ค่าคงที่

ค่าคงที่ (**Constants**) คือค่าของ **Literal** ใดๆ ที่ถูกกำหนดให้กับตัวแปรค่าคงที่ ค่าคงที่จะไม่สามารถเปลี่ยนแปลงได้ภายในโปรแกรมและเป็นได้เพียงค่าเดียวตั้งแต่โปรแกรมเริ่มจนถึงสิ้นสุดการทำงาน ในภาษา **PHP** คุณสามารถประกาศค่าคงที่ได้สองแบบโดยการใช้ฟังก์ชัน **define()** และคำสั่ง **Const**

```
define('NAME', 'MARCUSCODE');  
define('YEAR', 2016);  
define('blue', '#0000ff', true);
```

```
const PI = 3.14;  
const UNIT = "Inch";
```

# ฐานข้อมูล **MYSQL** เบื้องต้น

คู่มือการใช้งาน **Laravel** เพื่อทำ **Api** ในรูปแบบต่างๆ

# คำสั่ง SQL Command

Structured Query Language หรือ SQL คือคำสั่งบริหารจัดการฐานข้อมูล (Database) โดยเฉพาะ Relational Database Management System (RDBMS) เช่น Oracle, MySQL, Microsoft SQL Server, PostgreSQL, IBM DB2, Microsoft Access ซึ่งเราสามารถใช้ SQL command ในการสั่งการ หรือ จัดการข้อมูลภายในฐานข้อมูลเหล่านี้ได้

```
UPDATE clause [UPDATE country
SET clause [SET population = population + 1
WHERE clause [WHERE name = 'USA';
```

The diagram illustrates the structure of an SQL UPDATE statement. It shows three clauses: UPDATE clause, SET clause, and WHERE clause. The SET clause contains an expression 'population + 1'. The WHERE clause contains a predicate 'name = 'USA''. Brackets and labels identify these components: 'Expression' points to 'population + 1', 'Predicate' points to 'name = 'USA'', and 'Statement' points to the entire command structure.

# DATA Types

**SQL data types** เป็นการกำหนดชนิดของข้อมูลในตารางว่าเป็นข้อมูลแบบใด เช่น ข้อมูลตัวเลข, ตัวอักษร, วันเวลา หรือแบบไม่มีโครงสร้าง ซึ่งสิ่งเหล่านี้จำเป็นตั้งแต่เราเริ่มสร้าง **database table** เพื่อให้ข้อมูลที่เรากำลังใส่ลงสู่ **table** มีความถูกต้องตามที่วางเอาไว้ อีกทั้งยังช่วยให้ฐานข้อมูลหรือ **database** ของเราทำงานได้ง่ายขึ้นในการจัดเก็บ และการทำดรรชนี (**index**) ได้เหมาะสมกับข้อมูลที่เรากำลังใช้งาน โดย **data types** บน **database** มีด้วยกันหลายชนิด ขึ้นอยู่กับชนิดของฐานข้อมูล หรือ **database** ที่เราใช้งาน

DATA TYPE	DESCRIPT
Varchar(n)	Character string จำกัดจำนวนตัวอักษรไม่มากกว่า n ตัว
Int	-2,147,483,648 – 2,147,483,648
Datetime	Jan 1, 1970

# Operator

SQL Operator คือกระบวนการทำงานเพื่อเลือกข้อมูลด้วยเงื่อนไขหรือวิธีการที่ต้องการ โดยระบบฐานข้อมูล หรือ **Database** จะมีการจอง **key word** บางคำ หรือ ตัวอักษรไว้สำหรับให้ **SQL statement** ใช้ระบุเงื่อนไข (Where clause) เพื่อเป็นการสั่งระบบให้ทำตามเงื่อนไขที่วางไว้ เช่น การเปรียบเทียบ การคำนวณทางคณิตศาสตร์ การทำกระบวนการเหล่านี้มักจะใช้ระบุเงื่อนไขภายใต้ **SQL statement** และ สามารถกำหนดได้หลายเงื่อนไขภายใน **statement** นั้นๆ

OPERATOR	DESCRIP
คำนวณทางคณิต	+, -, *, /, %
เปรียบเทียบ	=, !=, <>, >, <, >=, <=, !<, !>
ตรรกศาสตร์	ALL, AND, ANY, BETWEEN, EXISTS, IN, LIKE, NOT, OR, IS NULL, UNIQUE

# Expression

SQL Expression คือ การรวมเอาค่าตัวแปรหรือข้อมูลตั้งแต่ 1 ตัวขึ้นไป รวมถึง [SQL Operator](#), SQL function ที่สามารถกำหนดค่าของข้อมูลได้ โดย SQL Expression คือ การเปรียบเสมือนสูตร หรือ ประโยค ในภาษาเขียน ซึ่งใช้ในการดึงข้อมูล (query) ทำให้เราสามารถเลือกข้อมูลที่เราต้องการจากฐานข้อมูล หรือ [database](#)

```
SELECT column1, column2, column  
FROM table_name  
WHERE [expression]
```

```
SELECT column1, column2, column  
FROM table_name  
WHERE [expression]  
AND [expression]
```



# Insert Query

SQL Insert into statement ข้อมูลที่ต้องระบุคือ table name, column name, value

```
INSERT INTO TABLE_NAME(column1, column2, columnN)
```

```
VALUES(value1, value2, value3, ...valueN)
```

# Select Query

SQL Select statement จำเป็นต้องใช้ข้อมูล **column name** และ **table name** ในการระบุชุดของข้อมูลที่จะแสดง

```
SELECT column1, column2, column
```

```
FROM table_name
```

```
SELECT * FROM table_name
```

# Update Query

การทำ SQL Update มีลักษณะคล้ายกับ [SQL Select statement](#) โดยเราต้องกำหนด **table** ที่ต้องการเปลี่ยนแปลงก่อน ตามด้วยค่าของข้อมูลใหม่แต่ละ **column** ซึ่งถ้ามีหลาย **column** เราสามารถใช้ **comma (,)** เป็นตัวขึ้นแต่ละ **column statement** รวมถึงสามารถนำเอา **Where condition** เพิ่มต่อเข้าไปได้เลย เพื่อให้ข้อมูลที่แสดงถูกกรองให้เหลือเฉพาะข้อมูลที่อยู่ในเงื่อนไขที่ต้องการ

UPDATE **table\_name**

SET **column1 = value1, column2 = value2**

WHERE [condition]

# Delete Query

การทำ SQL Delete query statement จะเป็นการลบข้อมูลทั้งบรรทัดภายใต้ table ที่กำหนด เราใช้ Where clause statement เป็นตัวระบุบรรทัดตามเงื่อนไขของผู้ใช้งาน มิเช่นนั้น ถ้าไม่ระบุ where clause ระบบจะถือว่าเป็นการลบข้อมูลทั้งหมดภายใต้ table นั้น

```
DELETE FROM table_name
```

```
WHERE [condition]
```

# GROUP BY

SQL GROUP BY มักจะใช้งานตามหลัง [SQL Select query statement](#) ที่อาจจะใช้ [SQL Where Clause](#) ร่วมด้วย และจะใช้หน้าหน้า [SQL ORDER BY](#) หากต้องการจัดเรียงลำดับข้อมูลหลายการจัดกลุ่มด้วย SQL GROUP BY

```
SELECT column1, column2
```

```
FROM table_name
```

```
WHERE [condition]
```

```
GROUP BY column1, column2
```

# ORDER BY

SQL ORDER BY Clause ที่ใช้งานภายใต้ SQL Select query statement เราสามารถใช้ 1 column หรือหลาย column ในการจัดเรียงข้อมูลก็ได้ และสามารถจัดเรียงจากน้อยไปมาก หรือ มากไปน้อยด้วย option

- ❑ ASC (Default) เรียงจาก น้อยไปมาก
- ❑ DESC เรียงจาก มากไปน้อย

```
SELECT column1, column2
```

```
FROM table_name
```

```
WHERE [condition]
```

```
ORDER BY column1 DESC
```

# INNER JOIN

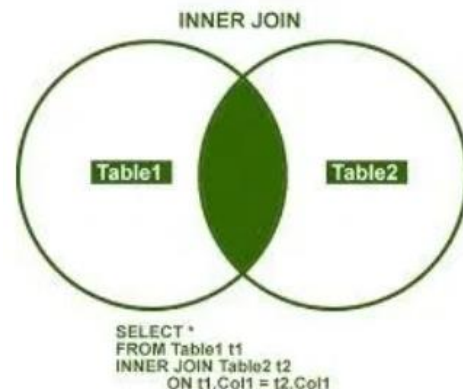
คำสั่ง **JOIN** ที่นิยมและเป็นค่า **default** คือ **INNER JOIN** ซึ่งคือการเชื่อมข้อมูลที่มีค่าทั้ง 2 **table** เหมือนกันในส่วน ของ **column** ที่กำหนดแต่ละ **table** หรือ บางครั้งจะเรียกว่า **EQUIJOIN** สำหรับบางฐานข้อมูล หรือ **database** หลักการของ **INNER JOIN** คือ สร้างข้อมูล **table** ใหม่จากข้อมูลของ 2 **table** (**table 1** และ **table 2**) โดยจะทำการเปรียบเทียบข้อมูลแต่ละบรรทัดของ **table 1** และ **table 2** เพื่อหาข้อมูล **column** ที่กำหนดทั้ง 2 **table** ที่ตรงกัน ซึ่งถ้าพบบรรทัดดังกล่าวแล้วก็จะเก็บไว้ในชุดข้อมูลที่จะแสดงผลในรูปแบบ **table** อันใหม่

```
SELECT table1.column1, table2.column2
```

```
FROM table1
```

```
INNER JOIN table2
```

```
ON table1.common_field = table2.common_field
```



# LEFT JOIN

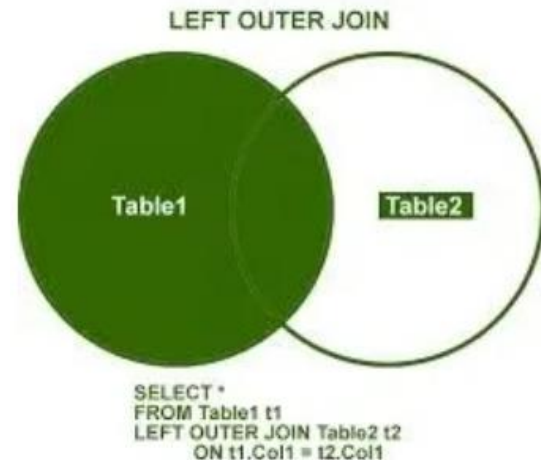
โดยปกติจะใช้สร้างความสัมพันธ์ของ **column** ระหว่าง **table** ที่เหมือนกัน ซึ่ง **SQL LEFT JOIN** ก็เช่นกัน แต่จะอาศัย **table** ด้านซ้าย หรือ **table** แรก เป็นข้อมูลตั้งต้นหลัก โดยข้อมูลบรรทัดที่ **table** ซ้ายมี แต่ **table** ขวา ไม่มี จะถือว่าเป็นข้อมูลใหม่ได้ด้วย โดยข้อมูลใหม่ที่ว่าจะมีแค่ข้อมูลจาก **table** ซ้ายเท่านั้นส่วน **column** ที่เกิดจาก **table** ขวา จะมีค่าเป็น **NULL** เพราะไม่มีข้อมูล แต่สำหรับบรรทัดที่มีเฉพาะ **table** ขวา แต่ไม่มีใน **table** ซ้าย จะถือว่าเป็นข้อมูลใหม่ เราสามารถเรียกอีกอย่างว่า “**LEFT OUTER JOIN**”

```
SELECT table1.column1, table2.column2
```

```
FROM table1
```

```
LEFT JOIN table2
```

```
ON table1.common_field = table2.common_field
```





# การทำ Index

- ❑ **SQL Index** คือ เครื่องมือค้นหาข้อมูลใน **table** ที่ช่วยให้ฐานข้อมูล หรือ **database** สามารถค้นหาข้อมูลได้รวดเร็วกว่าปกติ เพราะ **index** จะมีฐานข้อมูลที่จะระบุตำแหน่งของข้อมูลที่ต้องการ เปรียบเหมือนสารบัญที่ระบุเลขหน้าของหนังสือที่ค้นหา การนำเอา **index** มาใช้งานช่วยให้ **SQL Select** ที่ดึงข้อมูลด้วย **Where Clause** สามารถทำได้รวดเร็วยิ่งขึ้น แต่ก็จะทำให้ **SQL Update** และ **SQL Insert** ช้าลง เพราะทุกครั้งที่ทำการแก้ไข หรือเพื่อข้อมูล ระบบจำเป็นต้องทำการสร้าง **index** ให้ใหม่ด้วย
- ❑ การจะสร้าง **index** ให้กับ **table** จะใช้คำสั่ง **SQL CREATE INDEX statement** ซึ่งเราสามารถตั้งชื่อของ **index** รวมถึงระบุ **table** และ **column** ที่จะใช้ทำ **index** ได้ โดยสามารถกำหนดได้ว่าจะมีคุณสมบัติ **unique** ด้วยหรือไม่ (**UNIQUE constraint**) รวมถึงสามารถทำการ **index** โดยอาศัย **column** เดียว หรือ หลาย **column** ก็ได้

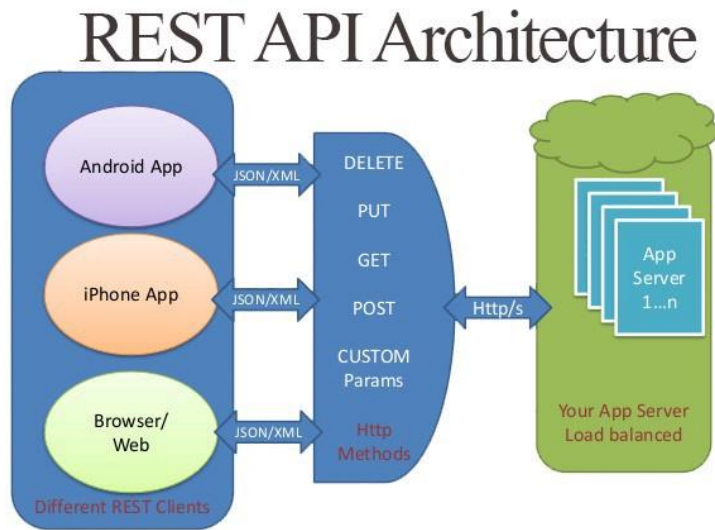
```
CREATE INDEX index_name on table_name(column_name);
```

# REST API

คู่มือการใช้งาน **Laravel** เพื่อทำ **Api** ในรูปแบบต่างๆ

# REST คืออะไร

REST ย่อมาจาก **Representational State Transfer** เป็นรูปแบบการส่งข้อมูลระหว่าง **Server-Client** รูปแบบหนึ่งซึ่งอยู่บนพื้นฐานของ **HTTP Protocol** เป็นการสร้าง **Web Service** เพื่อแลกเปลี่ยนข้อมูลกันผ่าน **Application** วิธีหนึ่งซึ่งส่งข้อมูลได้หลายชนิด ไม่ว่าจะเป็น **Text, XML, JSON** หรือส่งมาเป็นหน้า **HTML** เลยก็ยังมี แต่ส่วนใหญ่แล้วจะเลือกชนิด **JSON** กันซะมากกว่าด้วยความที่รองรับได้ทั้งหลายรูปแบบไม่ว่าจะเป็น **Browser** หรือ **Mobile** และยังสามารถใช้งานร่วมกับ **Web Service** ประเภทอื่นๆได้อีก เพียงแค่รู้ **URL** ก็สามารถแลกเปลี่ยนข้อมูลกันได้ รวมถึงยังจัดการง่ายกว่าด้วยเพียงแค่เรารับข้อมูลมา จากนั้นเราจะเอาข้อมูลไปแสดงผลยังไงก็ได้ตามใจเลย



# REST เบื้องต้น

ทำงานอยู่บนอยู่ใน HTTP Protocol ทำให้เวลาเราใช้งาน จะต้องอยู่บนพื้นฐาน HTTP Method เช่น GET, POST, PUT, DELETE เราจะใช้ Method ไหน เมื่อไร ก็ขึ้นอยู่กับว่า เราจะทำอะไรกับข้อมูล แต่ก็ต้องใช้คู่กับ Operation CURD เช่น เมื่อเราต้องการจะเรียกดูข้อมูลทั้งหมดก็ใช้ GET เมื่อเราต้องการเพิ่มข้อมูลก็ใช้ POST ดังตัวอย่างเบื้องต้น ซึ่งจริงๆมีอีกหลายตัวดูได้

*GET — R(etrieve) เรียกดูข้อมูล*  
*POST — C(reate) เพิ่มข้อมูล*  
*PUT — U(pdate) แก้ไขข้อมูล*  
*DELETE — D(elete) ลบข้อมูล*

# อ้างอิงจาก

ขอบคุณสำหรับข้อมูลดี ๆ

- หาข้อมูล **mysql** เพิ่มเติมได้ที่ <https://saixiii.com/sql-command>
- หาข้อมูล **php** เพิ่มเติมได้ที่ <http://marcuscode.com/lang/php>
- เพิ่มเติมได้ที่ <https://medium.com/@settawatjanpuk>

❑ `xampp\htdocs>composer create-project --prefer-dist laravel/laravel:^7.0 blog`

```
Select Command Prompt
C:\>xampp>cd htdocs>
The syntax of the command is incorrect.

C:\>xampp>cd htdocs
Access is denied.

C:\>xampp\htdocs
'xampp' is not recognized as an internal or external command,
operable program or batch file.

C:\>cd xampp
C:\xampp>cd htdocs
C:\xampp\htdocs>composer creat-project --prefer-dist laravel/laravel:^7.0 blog
C:\xampp\htdocs>composer creat-project --prefer-dist laravel/laravel:^7.0 gpblog

[Symfony\Component\Console\Exception\CommandNotFoundException]
Command "creat-project" is not defined.

Did you mean this?
    create-project

C:\xampp\htdocs>
```

```
Windows PowerShell
PS D:\xampp\htdocs\gpblog> php artisan serve
[info] development server started: http://127.0.0.1:8000
[Tue Sep 14 10:16:59 2021] PHP 7.4.23 Development Server (http://127.0.0.1:8000) started
[Tue Sep 14 10:17:35 2021] 127.0.0.1:52489 Accepted
[Tue Sep 14 10:17:35 2021] 127.0.0.1:61134 Accepted
[Tue Sep 14 10:17:36 2021] 127.0.0.1:52489 Closing
[Tue Sep 14 10:17:37 2021] 127.0.0.1:61134 [200]: GET /favicon.ico
[Tue Sep 14 10:17:37 2021] 127.0.0.1:61134 Closing
PS D:\xampp\htdocs\gpblog>
```

□ php artisan serve