# OOP in PHP ตอนที่ 1

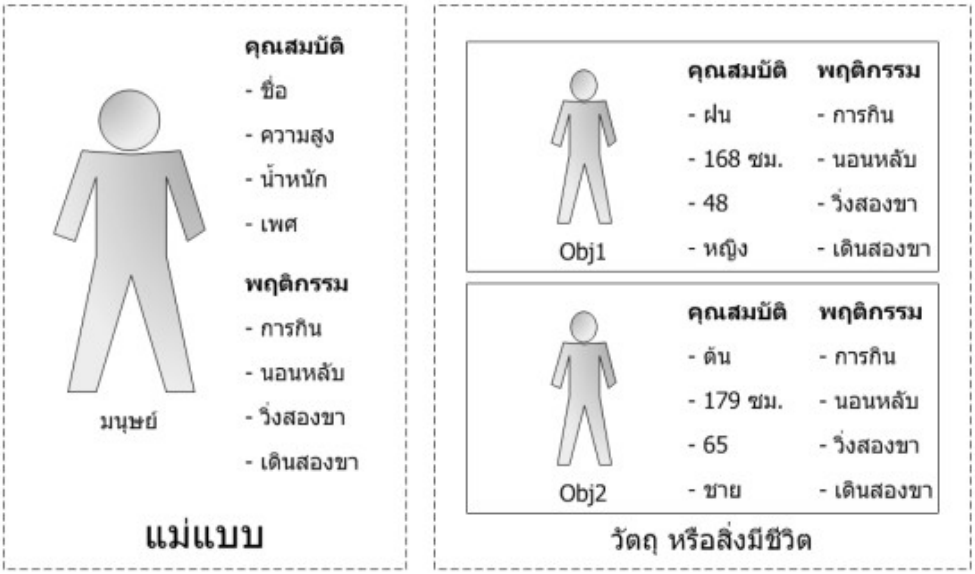
สวัสดีครับ ในบทความนี้เราจะกล่าวถึงการนำเอาภาษาโปรแกรม **PHP** มาเขียนในรูปแบบของ **Object Oriented Programming** โดยในบทความนี้เราจะไม่กล่าวถึงพื้นฐานของภาษา PHP มากนัก โดยบทความนี้จะเหมาะกับผู้ที่เขียนและใช้ภาษาโปรแกรม PHP มาบ้างแล้ว โดยถ้าใช้อยู่ในระดับการใช้ Function มาก่อนน่าจะช่วยทำให้เข้าใจและทำให้เห็นภาพได้มากขึ้นครับ

## มาสร้างโลกน้อย ๆ ในการเขียนโปรแกรมกันดีกว่า

ก่อนอื่นเราต้องปรับความคิดในการเขียน PHP จากการเขียนแบบลำดับขั้นจากบนลงล่าง (Structural หรือ Procedural Programming) มาเป็นการเขียนแบบเชิงวัตถุ (Object Oriented Programming) เสียก่อน

โดยการเขียนโปรแกรมเชิงวัตถุนั้น เป็นการเขียนโปรแกรมให้เข้าใกล้การดำเนินไปตามสภาพแวดล้อมที่เป็นจริงของโลกเรา กล่าวคือ เรามองทุกอย่างในการเขียนโปรแกรมเป็นดั่งการดำรงอยู่ของสิ่งต่าง ๆ บนโลกนี้นั้นเอง โดยทุก ๆ สิ่งบนโลกนี้มักเกิดจากแม่แบบ (**Class**) แทบทั้งสิ้น ซึ่งแม่แบบเหล่านี้นั้นจะมีสิ่งที่จำเป็นอยู่ 2 ส่วนคือ คุณสมบัติ (**Properties**) และแบบแผนพฤติกรรม (**Method Behavior หรือ Method**) อาจจะมีความเหมือนหรือต่างกันทั้งในคุณสมบัติและพฤติกรรม โดยผลของแบบแผนพฤติกรรมนั้นจะได้รับเป็นแบบใด ขึ้นอยู่กับคุณสมบัติในช่วงเวลานั้น ๆ ซึ่งจะอธิบายให้เห็นภาพชัดเจนได้ง่ายขึ้นดังตัวอย่างด้านล่าง

ผู้เขียนจะขอยกตัวอย่างที่ง่ายก่อน โดยอยากให้มองถึงคำว่า **“มนุษย์”** เป็นแม่แบบ ทุก ๆ คนบนโลกนี้เกิดจากแม่แบบ (Class) “มนุษย์” ซึ่งวัตถุ (**Object**) ที่เกิดจากแม่แบบ โดยคุณสมบัติ (Properties) ก็จะชื่อเรียกแตกต่างกันไป, มีความสูง, น้ำหนัก, เพศ และส่วนประกอบทางร่างกายที่แตกต่างกัน ซึ่งสิ่งเหล่านี้คือคุณสมบัติ (Properties) ของ “มนุษย์” โดยจะมีพฤติกรรม (Method) ที่เหมือน ๆ กันโดยพื้นฐานของ “มนุษย์” ซึ่งได้แก่ การกิน, นอนหลับ, วิ่งและเดินสองขา ฯลฯ ซึ่งถือเป็นพฤติกรรมพื้นฐานของแม่แบบนี้ ถ้าเราเขียนเป็นภาพให้เห็นชัดเจนขึ้นก็จะได้ดังรูปนี้




ภาพแสดงแม่แบบ และวัตถุ

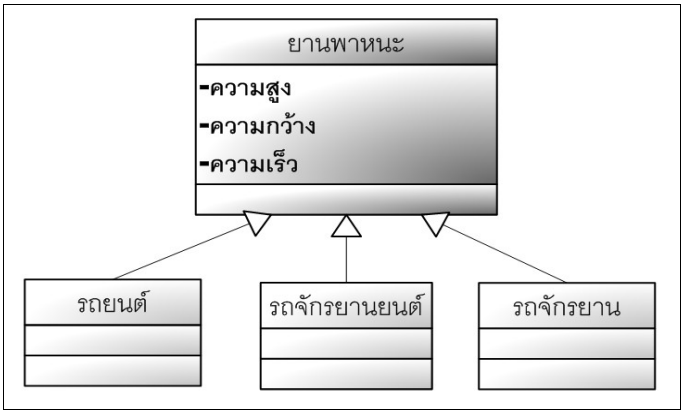
จากรูปเราจะเห็นว่า “**Obj1**”และ “**Obj2**” นั้นมีรูปแบบพฤติกรรมและคุณสมบัติเหมือนกันแม่แบบ **มนุษย์** แต่คุณสมบัติของทั้งสองวัตถุนั้นได้ถูกกำหนดต่อมาให้แตกต่างกัน ทำให้ผลของพฤติกรรมนั้นแตกต่างกันไปตามข้อมูลของคุณสมบัติที่เปลี่ยนแปลงไป เช่น Obj1 นั้นมีความสูง 168 ซม. , น้ำหนัก 48 กก. และเป็นเพศหญิง ซึ่งมีคุณสมบัติแตกต่างกับ Obj2 ที่มีความสูง 179 ซม. , น้ำหนัก 65 กก. และเป็นเพศชาย กล่าวคือในสภาวะเป็นจริงแล้ว **ความแตกต่างของทั้งของวัตถุทั้งสองนั้นถูกกำหนดจากคุณสมบัติ ซึ่งทำให้การผลของพฤติกรรมนั้นแตกต่างกัน แม้จะเกิดจากแม่แบบเดียวกับก็ตาม**

**สรุปว่า** “แม่แบบ (Class) มีความเป็น**นามธรรม (Abstract)** และวัตถุ (Object) มีความเป็น**รูปธรรม (Concrete)**”

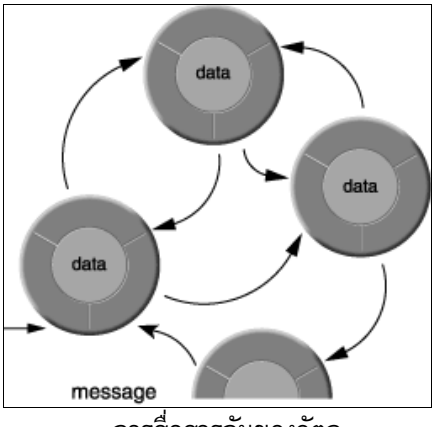
โดยคุณสมบัติต่าง ๆ ของวัตถุนั้นต้องมีการห่อหุ้มอยู่ภายใน (**Encapsulation** ) ซึ่งทำให้คุณสมบัติของข้อมูลนั้นถูกต้องเสมอ โดยจากตัวอย่างที่แล้วนั้นในกรณีของคุณสมบัติของเพศนั้น จะต้องอยู่ภายใต้เงื่อนไขของเพศชาย และหญิงเท่านั้น จึงจำเป็นต้องมีตัวควบคุมคุณสมบัติ หรือแบบแผนพฤติกรรม เข้ามาเกี่ยวข้องซึ่งจะได้กล่าวต่อไป โดยความถูกต้องของข้อมูลภายในนั้นสำคัญมาก เพราะนั่นหมายถึงการแสดงคุณสมบัติที่ถูกต้องของวัตถุนั้นๆ ด้วย รวมถึงถ้าเรามองในด้านการเขียนโปรแกรมแล้ว การห่อหุ้มภายในนั้นช่วยให้การดูแลรักษาและพัฒนาชุดคำสั่งนั้นทำได้ง่ายขึ้น เพราะชุดคำสั่งได้ถูกแยกออกเป็นส่วน ๆ และยังบอกถึงพฤติกรรมของสิ่งที่ห่อหุ้มด้วยว่าควรเป็นพฤติกรรมอะไร และเมื่อเราพบพฤติกรรมนั้น ๆ แล้วผลของพฤติกรรมนั้นจะให้ข้อมูลและผลการทำงานออกมาเป็นอะไรด้วย

ภาพแสดงการห่อหุ้มภายในคุณสมบัติ

คุณสมบัติที่ทำให้การเขียนโปรแกรมเชิงวัตถุนั้นได้รับความนิยมมาก คือการสืบทอด (**Inheritance**) ของแม่แบบ (Class) ที่ทำให้ลูกแบบ (**Sub-Class**) นั้นมีความเฉพาะเจาะจงมากขึ้นไปอีกไปอีก และยังช่วยให้การสร้างสรรค์งานนั้นทำได้ง่ายเพราะเราไม่จำเป็นต้องเขียนโปรแกรมหลาย ๆ อย่างซ้ำ ๆ กันในแม่แบบที่มีคุณสมบัติเหมือน ๆ กัน เช่นแม่แบบ “ยานพาหนะ” มีลูกแบบ ชื่อได้แก่ รถยนต์, รถจักรยานยนต์ และจักรยาน เป็นต้น ซึ่งทั้ง 3 ลูกแบบที่กล่าวมานั้นมีคุณสมบัติบางอย่างที่เหมือนกันจนได้เป็นยานพาหนะ เช่นความสูง, ความกว้าง และความเร็วในการเคลื่อนที่เป็นต้น ซึ่งการแบ่งเป็นลูกแบบนี้ทำให้ความเฉพาะเจาะจงของลูกแบบมีมากขึ้นทำให้การสร้างวัตถุใด ๆ จากลูกแบบเหล่านี้นั้นมีคุณสมบัติที่เที่ยงตรงต่อการแสดงพฤติกรรมได้มากขึ้นด้วย และลดการซับซ้อนของโครงสร้างแม่แบบลงได้มาก เพราะได้ถูกกำหนดคุณสมบัติหลัก ๆ ไว้แล้วในแม่แบบ ทำให้ลูกแบบมีความซับซ้อนน้อยลง



การสืบทอดคุณสมบัติ



การสื่อสารกันของวัตถุ

ในการส่งข้อมูลระหว่างวัตถุนั้นเราเรียกว่าการส่งข้อความ (Messaging) ซึ่งข้อความ (Message) นั้นจะถูกส่งผ่านแบบแผนพฤติกรรม (Method) เพื่อตรวจสอบความถูกต้องของข้อความที่จะเข้าหรือออกจากวัตถุ (Object) ก่อนที่จะทำการประมวลข้อความนั้นเพื่อแก้ไขข้อมูลหรือคุณสมบัติ (Properties) ภายในวัตถุ ซึ่งการตรวจสอบนี้เป็นไปตามหลักการห่อหุ้มข้อมูลภายใน เพื่อป้องกันไม่ให้วัตถุทำงานผิดไปจากพฤติกรรมที่กำหนด โดยขอยกตัวอย่าง ที่เข้าใกล้กับชีวิตประจำวัน โดยเรามีนาย ก. และนาย ข. ซึ่งเกิดจากแม่แบบ มนุษย์ ซึ่งนาย ก. ได้ทำการถาม อายุนาย ข. โดยที่นาย ข. นั้นจะตอบอายุกลับไปก็ได้หรือไม่ ขึ้นอยู่กับคุณสมบัติบางประการของวัตถุนั้น ๆ เช่นเพศ เป็นต้น โดยที่ **นาย ข. เป็นเพศชาย** อาจจะตอบกลับไปในทันที **แต่ในกรณีที่เป็นเพศหญิง**อาจจะไม่ตอบกลับข้อความนี้กลับไป ซึ่งจากตัวอย่างดังกล่าวจะเห็นได้ว่าการส่งข้อความไป และกลับนั้นจะได้ผลตอบรับอย่างไรนั้น ขึ้นอยู่กับคุณสมบัติ ณ. เวลาใดเวลาหนึ่งของวัตถุในเวลานั้น ๆ และวัตถุแต่ละวัตถุนั้นจะมีความเป็นอิสระกันของคุณสมบัติ เหมือนดังตัวอย่างที่ นาย ก. และนาย ข. ไม่ได้เป็นคนเดียวกันนั้นเอง

โดยยังมีคุณสมบัติการกระทำหลายรูปแบบ (**Polymorphism**) ซึ่งจะไม่ขอผู้ถึงในตอนนี้ เพราะอาจทำให้งงได้ ผู้เขียนจะขอพูดถึงในระดับที่สูงต่อไป

จากที่เราได้ศึกษาจากข้อมูลด้านบนนั้นเราได้สรุปส่วนที่สำคัญในการพัฒนาของ Object Oriented Programming ไว้ดังต่อไปนี้

- แม่แบบ (Class)
- ลูกแบบ (Sub-Class)
- วัตถุ (Object)
- การส่งข้อความ (Messaging)
- ข้อความ (Message)
- การห่อหุ้มข้อมูล(Encapsulation)
- การสืบทอด (Inheritance)
- การทำหลายรูปแบบ (Polymorphism)
- คุณสมบัติ (Properties)
- แบบแผนพฤติกรรม (Method Behavior)

# “สวัสดีชาวโลก” กับ PHP บนโลกแห่งความจริง

> ในบทความนี้ทางผู้เขียนจะอ้างอิงกับ PHP Version 4 ก่อน เพราะมีข้อกำหนดที่ไม่ยุ่งยาก และเหมาะแก่การนำมาศึกษามากกว่า PHP Version 5

โดยในการเขียนแบบโปรแกรมเชิงวัตถุใน PHP นั้นไม่ยาก โดยคำที่ใช้เป็นคำหลัก (Keyword) ในการเขียนได้แก่ **class, function, this** และ **parent** เป็นต้น

จากตัวอย่างในช่วงแรกผู้เขียนจะสร้าง แม่แบบ (Class) **human** ขึ้นมาก่อน โดยวิธีการสร้างแม่แบบใน PHP มีดังนี้

```
<?php
class classname {

}
?>
```

รูปแบบคำสั่งที่ 1

1. คำหลักที่เราจะใช้นั้นคือ ***class***
2. หลังคำหลัก ***class*** ตามด้วยชื่อแม่แบบ (**classname**)
3. เปิดและปิดด้วยเครื่องหมายปีกกา เพื่อกำหนดช่วงการเขียนชุดคำสั่งของแม่แบบ

เมื่อเรารู้รูปแบบการสร้างแม่แบบแล้ว เราจะมาสร้างแม่แบบเพื่อใช้งานจริงกันเลย

```
<?php
class human {

}
?>
```

ชุดคำสั่งที่ 1

จากชุดคำสั่ง (**Source Code**) ที่ 1 นี้เป็นเพียงการประกาศแม่แบบเท่านั้น ซึ่งไม่สามารถทำงานได้

```
<?php
class human {
   function Talk(){
      echo "Hello World";
   }
}
?>
```

ชุดคำสั่งที่ 2

ทำให้เกิดชุดคำสั่งที่ 2 ซึ่งเราได้เพิ่มพฤติกรรม (Method) ลงไป โดยผมขอยกตัวอย่างว่าให้วัตถุใด ๆ ที่สร้างจากแม่แบบ human นั้นสามารถพิมพ์คำว่า "Hello World" ออกมาได้โดยใช้ผ่านพฤติกรรมที่ชื่อว่า "Talk" มาเป็นหัวข้อตัวอย่าง โดยการเขียนพฤติกรรมในแม่แบบนั้นก็เขียนรูปแบบเดียวกับฟังก์ชันทั่ว ๆ ไปที่เราคุ้นเคยกันใน PHP แบบเดิม ๆ นั่นเอง โดยใช้คำหลักคือ ***function*** และตามด้วยชื่อพฤติกรรม ซึ่งการประกาศ function ในแม่แบบนี้ก็เหมือนกับ function โดยทั่วไปนั้นเอง โดยจากตัวอย่างชุดคำสั่งที่ 2 นี้เราจะสร้างวัตถุจากแม่แบบที่เขียนขึ้นมาทำงาน โดยใช้คำหลัก ***new*** โดยรูปแบบการสร้างวัตถุขึ้นมานั้นมีดังนี้

```
$objectname = new classname();
```

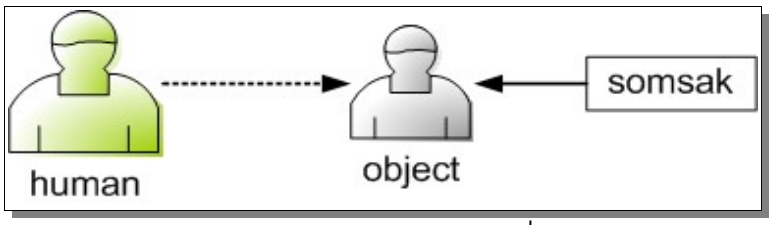
รูปแบบคำสั่งที่ 2

จากรูปแบบคำสั่งที่ 2 นี้จะเห็นรูปแบบการสร้างวัตถุโดยที่ ***$objectname*** นี้เป็นชื่อของ**ชื่ออ้างอิงวัตถุ** หรือเรียกว่า**ตัวแปรอ้างอิง**ก็ได้ โดยที่ตัวแปรนี้จะไปอ้างอิงกับวัตถุที่เราสร้างขึ้นในหน่วยความจำอีกที โดยหลักการตั้งชื่อตัวแปรอ้างอิงกับวัตถุนั้นจะเหมือนกับตัวแปรทั่วไปทุกประการ

```
<?php
class human {
     function Talk(){
          echo "Hello World";
     }
}
$somsak = new human();
$somsak->Talk();
?>
```

ชุดคำสั่งที่ 3

โดยจากชุดคำสั่งที่ 3 เราจะได้วัตถุที่มีตัวแปรอ้างอิงที่ชื่อว่า "somsak" เมื่อเรามาดูตัวอย่างความสัมพันธ์ระหว่างแม่แบบ, ตัวแปรอ้างอิงวัตถุ และวัตถุ ได้จากภาพด้านล่างนี้ ว่ามีความสัมพันธ์กันอย่างไร



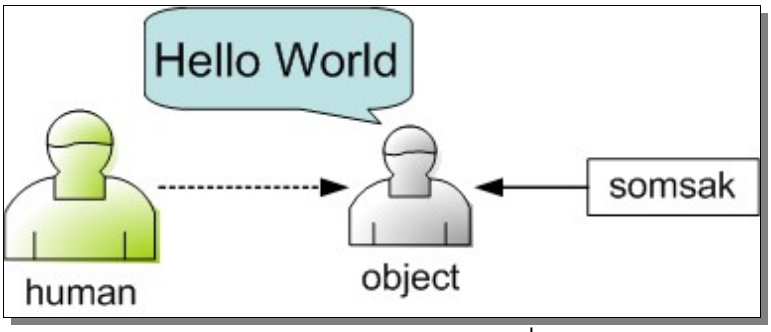

ภาพแสดงความสัมพันธ์ที่ 1

จากภาพแสดงความสัมพันธ์ที่ 1 นี้จะเห็นว่ามีวัตถุอยู่ 1 วัตถุอยู่ในหน่วยความจำโดยมีตัวแปรอ้างอิงกับวัตถุนี้ชื่อว่า somsak อีกที โดยที่วัตถุนี้เกิดจากแม่แบบ human นั้นเอง ซึ่งจากชุดคำสั่งสั่งดังกล่าวนั้นก็มีการให้พิมพ์คำว่า Hello World ออกมา ดัง “ผลดังผลการทำงานของชุดคำสั่งที่ 3”

```
Hello World
```

ผลการทำงานของชุดคำสั่งที่ 3

โดยเราจะสังเกตเห็นเครื่องหมาย**อ้างอิง -> (Reference/Arrow Operator**) ซึ่งเป็นการชี้ไปยังคุณสมบัติภายในหรือเรียกใช้พฤติกรรมใด ๆ ในวัตถุนั้น ๆ โดยดูได้จากชุดคำสั่งที่ 3 ที่ตัวแปรอ้างอิงวัตถุ somsak เพื่อเรียกใช้พฤติกรรม **Talk** ขึ้นมานั่นเองซึ่งจากชุดคำสั่งที่ 3 นี้เราจะมาวาดภาพเป็นแม่แบบ และวัตถุกัน เพื่อให้เราจะได้เชื่อมโยงความคิดระหว่างชุดคำสั่งและแนวคิดการเขียนโปรแกรมเชิงวัตถุ




ภาพแสดงความสัมพันธ์ที่ 2

โดยจากภาพแสดงความสัมพันธ์ที่ 2 นี้เปรียบเทียบกับชุดคำสั่งที่ 3 นั้นเราจะเห็นว่าตัวแปรอ้างอิงวัตถุ somsak นั้นไปอ้างอิงกับวัตถุที่เกิดจากแม่แบบ human ซึ่งและเมื่อเกิดขึ้นมาแล้วนั้น ตัวแปรอ้างอิงวัตถุก็ทำการสั่งเรียกใช้พฤติกรรม “Talk” โดยภาพนี้จะเห็นว่าวัตถุที่ตัวแปรอ้างอิง somsak ไปอ้างอิงอยู่ และได้สั่งให้พฤติกรรม Talk ทำงานขึ้นโดยพิมพ์คำว่า “Hello World” ออกมา

## มาเริ่มเขียนตัวละครฉบับ PHP กันดีกว่า

จากชุดคำสั่งที่ 1-3 และภาพแสดงความสัมพันธ์ทั้งสองรูปที่ได้แสดงไปแล้วนั้น จะเห็นได้ว่าเมื่อเราเขียนโปรแกรมแบบเชิงวัตถุก็เหมือนกับเราเขียนบทละครให้แต่ละวัตถุทำสิ่งใด ๆ ก็ตามที่เราต้องการ โดยกำหนดตัวแปรอ้างอิงหรือความหมายหนึ่งก็คือการสร้างชื่อนักแสดงอีกทีหนึ่งเพื่อง่ายต่อการจัดการคุณสมบัติและแสดงพฤติกรรมให้ตรงกับวัตถุนั้น ๆ โดยตัวอย่างที่ได้กล่าวไปนั้นอาจจะยังไม่เห็นภาพมากนัก เรามาดูต่อว่าเมื่อแม่แบบนั้นมีการกำหนดคุณสมบัติเพิ่มเข้ามาจะทำให้ผลของพฤติกรรมเปลี่ยนไปและแตกต่างกัน ซึ่งเราจะมาดูว่าทำให้เกิดความแตกต่างกันได้อย่างไร โดยเรากำหนดคุณสมบัติและพฤติกรรมดังต่อไปนี้ ชื่อแม่แบบคือ **human** โดยที่แม่แบบ human มี **คุณสมบัติ (Properties)** คือ **ชื่อ, ความสูง, น้ำหนัก, อายุ และเพศ** และมี**พฤติกรรม (Method)** คือ **บอกชื่อ, บอกความสูง, บอกน้ำหนัก และบอกเพศ** ได้

เมื่อได้รูปแบบที่ต้องการแล้วเราก็จะมาเขียนชุดคำสั่งกันเลย

```
<?php
class human {
   var $name = "human";// default name = "human"
   var $sex = 1;     // default = 1 and 1 = male, 2 = female
   var $height = 0;     // height > 0
   var $weight = 0;     // weight > 0
   var $age    = 0;     // age > 0
   function showName(){
      return $this->name;
   }
   function showSex(){
      return $this->sex;
   }
   function showHeight(){
      return $this->height;
   }
   function showWeight(){
      return ($this->sex == 2 ? 'No Answer' : $this->weight);
   }
   function showAge(){
      return ($this->sex == 2 ? 'No Answer' : $this->age);
   }
}
?>
```

ชุดคำสั่งที่ 4

จากชุดคำสั่งที่ 4 นี้ เราได้พบคำหลักอีกคำคือ ***var*** โดยคำนี้เป็นคำหลักที่ใช้กำหนดคุณสมบัติเริ่มต้นของแม่แบบ ซึ่งเป็น ดังข้อกำหนดว่าเมื่อสร้างวัตถุขึ้นแล้วคุณสมบัติดังกล่าวจะติดตัวไปกับวัตถุนั้น ๆ ด้วย ซึ่งคุณสมบัตินั้นจะแยกอิสระออกจากวัตถุอื่น ๆ โดยรูปแบบการเขียนคุณสมบัติก็คือ

```
var $propertyname;
```

รูปแบบคำสั่งที่ 3

โดยที่หลังคำหลัก ***var*** จะตามด้วยชื่อคุณสมบัติ **$propertyname** ซึ่งการตั้งชื่อคุณสมบัตินั้นก็ใช้หลักการตั้งชื่อแบบเดียวกับตัวแปรโดยทั่วไป

โดยการเรียกใช้คุณสมบัติ และพฤติกรรมภายในแม่แบบเดียวกันนั้นจะใช้ผ่านคำหลักที่ชื่อ ***this*** ซึ่งเราจะทำการอ้างอิงกับชื่อคุณสมบัติหรือพฤติกรรมอีกทีหนึ่งโดยใช้เครื่องหมายอ้างอิง **->** เข้ามาร่วมการทำงานโดยวิธีการเขียนคือ

```
$this->propertyname;
$this->methodname(<parameter*>);
```

รูปแบบคำสั่งที่ 4

โดยที่ ***$this*** มีความหมายว่า "**การเชื่อมโยงคุณสมบัติ และพฤติกรรมของตนเอง**" ซึ่งในรูปแบบคำสั่งที่ 4 นี้ก็ได้มีตัวอย่างการอ้างอิงถึงพฤติกรรม ซึ่งพฤติกรรมนี้เราสามารถเรียกใช้ได้เหมือน ๆ กับฟังก์ชันทั่วๆ ไปและรวมไปถึงการส่งพารามิเตอร์ต่าง ๆ เข้าไปได้ด้วย

จากตัวชุดคำสั่งที่ 4 เรานำมาสร้างวัตถุเพื่อพิสูทธิ์ว่าวัตถุนั้น ๆ มีความเป็นอิสระต่อกัน โดยเราจะสร้างวัตถุ 2 วัตถุที่มีชื่ออ้างอิงว่า somsak และ somying โดยที่กำหนดให้ชื่อ Somsak เป็นผู้ชาย มีอายุ 28 ปี, ความสูง 178 ซม. น้ำ

หนัก 65 กก. และ somying เป็นผู้หญิง มีอายุ 25 ปี, ความสูง 168 ซม. นำหนัก 48 กก. เมื่อกำหนดเสร็จแล้ว เราจะบอกให้วัตถุทั้งสองบอกข้อมูลของตัวเองผ่านพฤติกรรมที่ได้กำหนดไว้ในชุดคำสั่งที่ 4 ที่ได้เขียนไว้ก่อนหน้านี้ ได้ดังนี้

```
$somsak = new human();
$somsak->name = 'Somsak';
$somsak->sex = 1;
$somsak->height = 178;
$somsak->weight = 65;
$somsak->age = 28;
echo 'Name : '.$somsak->showName().'<br />';
echo 'Sex : '.$somsak->showSex().'<br />';
echo 'Height : '.$somsak->showHeight().'<br />';
echo 'Weight : '.$somsak->showWeight().'<br />';
echo 'Age : '.$somsak->showAge().'<br />';

$somying = new human();
$somying->name = 'Somying';
$somying->sex = 2;
$somying->height = 168;
$somying->weight = 48;
$somying->age = 25;
echo 'Name : '.$somying->showName().'<br />';
echo 'Sex : '.$somying->showSex().'<br />';
echo 'Height : '.$somying->showHeight().'<br />';
echo 'Weight : '.$somying->showWeight().'<br />';
echo 'Age : '.$somying->showAge().'<br />';
```

ชุดคำสั่งที่ 5

จากตัวอย่างชุดคำสั่งดังกล่าวนี้เราจะได้คำตอบของผลการทำงานดังต่อไปนี้

```
Name : Somsak
Sex : male
Height : 178
Weight : 65
Age : 28
Name : Somying
Sex : female
Height : 168
Weight : No Answer
Age : No Answer
```

ผลการทำงานของชุดคำสั่งที่ 5

จากชุดคำสั่งที่ 5 และผลการทำงานนี้บอกเราว่าวัตถุที่สร้างจากแม่แบบ human นั้นเมื่อถูกกำหนดคุณสมบัติว่าเป็นเพศชายแล้วถ้าเราสั่งให้แสดงอายุและน้ำหนักก็จะแสดงออกมา แต่ถ้าถูกกำหนดคุณสมบัติเป็นเพศหญิงแล้วกลับไม่แสดงอายุและน้ำหนักออกมาให้เรา ซึ่งเกิดจากการที่เราได้กำหนดในพฤติกรรมนี้ในแม่แบบไว้แล้วว่า โดยอยู่ในพฤติกรรมที่ชื่อ showAge และ showWeight ว่าถ้าเป็นเพศหญิงจะ No Answer ออกมา ด้วยเหตุผลบางประการ ;P

จากตัวอย่างนี้ยังมีจุดอ่อนเรื่องของการตั้งค่าคุณสมบัติหลายอย่าง เช่นการกำหนดเพศนั้นจะต้องเป็น 1 ที่เป็นเพศชายและ 2 ที่เป็นเพศหญิงซึ่งถ้าเราตั้งค่าคุณสมบัติโดยตรงแบบชุดคำสั่งที่ 5 นั้นจะทำให้ข้อมูลคุณสมบัติผิดเพี้ยนไปจากที่ควรจะเป็นเราจึงจำเป็นต้องปรับเปลี่ยนการตั้งค่าคุณสมบัติใหม่โดยทำงานผ่านพฤติกรรมแทนซึ่งช่วยในการกรองข้อมูลให้เราก่อนทำการตั้งค่าให้กับคุณสมบัติจริง ๆ ซึ่งเป็นคุณสมบัติที่อยู่ในข้อกำหนดการห่อหุ้มภายใน (Encapsulation) นั้นเอง โดยตามข้อกำหนดที่รู้โดยทั่วกันในวงการการเขียนโปรแกรมเชิงวัตถุว่าการตั้งค่าให้กับ

คุณสมบัตินั้นต้องใช้คำขึ้นต้นว่า “***set***” ที่แปลว่า **ตั้งค่า** และการเรียกใช้ค่าคุณสมบัตินั้นใช้คำขึ้นต้นว่า “***get***” ที่แปลว่า **เข้าถึง** แล้วตามด้วยชื่อคุณสมบัตินั้น ๆ เพื่อง่ายต่อการเรียกใช้และสื่อความหมายมากขึ้น ทางผู้เขียนจึงขอนำชุดคำสั่งที่ 5 มาปรับปรุงให้ดีขึ้นไปอีก

```
<?php
class human {
    var $name = "human";// default name = "human"
    var $Sex= 1;           // default = 1 and 1 = male, 2 = female
    var $height  = 0;  // height > 0
    var $weight  = 0;  // weight > 0
    var $age     = 0;  // age > 0
    function setName($value = 'human'){
        $this->name = $value;
    }
    function setSex($value = 1){
        $this->Sex = ($value == 1 || $value == 2 ? $value : 1);
    }
    function setHeight($value = 1){
        $this->height = ($value > 0 ? $value : 1 );
    }
    function setWeight($value = 1){
        $this->weight = ($value > 0 ? $value : 1 );
    }
    function setAge($value = 1){
        $this->age = ($value > 0 ? $value : 1 );
    }
    function getName(){
        return $this->name;
    }
    function getSex(){
        return ($this->Sex == 1 ? 'male' : 'female');
    }
    function getHeight(){
        return $this->height;
    }
    function getWeight(){
        return ($this->Sex == 2 ? 'No Answer' : $this->weight);
    }
    function getAge(){
        return ($this->Sex == 2 ? 'No Answer' : $this->age);
    }
}
?>
```

ชุดคำสั่งที่ 6

จากชุดคำสั่งที่ 6 จะเห็นได้ว่า ผู้เขียนนั้นได้ปรับปรุงจากการใช้คำว่า *show* เป็นคำว่า *get* แทน **และได้เพิ่มพฤติกรรมใหม่ที่มี *set* เป็นคำขึ้นต้นและตามด้วยชื่อตัวแปร โดยสร้างขึ้นมาเพื่อให้ใช้พฤติกรรมนี้ในการเปลี่ยนแปลงค่าแทนการตั้งค่าให้กับคุณสมบัติโดยตรง**แบบชุดคำสั่งที่ 5 โดยในพฤติกรรม setName, setSex, setHeight, setWeight และ setAge โดยจะมีค่าเริ่มต้นของการทำงานไว้ที่พารามิเตอร์เป็น 1 ไว้ก่อนแล้วเพื่อป้องกันผู้เขียนโปรแกรมนั้นลืมใส่ค่าลงไป ซึ่งใน setName นั้นได้กำหนดพารามิเตอร์ตั้งต้นไว้เช่นกัน แต่เป็นการกำหนดชื่อ human แทน ซึ่งใน setSex นั้นได้มีการดักจับค่าว่าต้องมีค่าเพียง 1 และ 2 เท่านั้น ส่วน setHeight, setWeight และ setAge ก็ได้ดักจับค่าว่า ค่าที่ใส่เข้ามานั้นต้องมีค่ามากกว่า 0 เสมอ ซึ่งจากที่เราได้ทำแบบนี้ทำให้การตั้งค่าและเข้าถึงคุณสมบัติต่าง ๆ มีความถูกต้องของข้อมูลมากขึ้นกว่าเดิม เมื่อเรานำไปใช้งานก็จะมีรูปแบบดังชุดคำสั่งที่ 7 นี้

```
$somsak = new human();
$somsak->setName('Somsak');
$somsak->setSex(); <---- ไม่ได้ใส่ค่าลงไป
$somsak->setHeight(178);
$somsak->setWeight(65);
$somsak->setAge(28);
echo 'Name : '.$somsak->getName().'<br />';
echo 'Sex : '.$somsak->getSex().'<br />';
echo 'Height : '.$somsak->getHeight().'<br />';
echo 'Weight : '.$somsak->getWeight().'<br />';
echo 'Age : '.$somsak->getAge().'<br />';

$somying = new human();
$somying->setName('Somying');
$somying->setSex(2);
$somying->setHeight(168);
$somying->setWeight(48);
$somying->setAge(25);
echo 'Name : '.$somying->getName().'<br />';
echo 'Sex : '.$somying->getSex().'<br />';
echo 'Height : '.$somying->getHeight().'<br />';
echo 'Weight : '.$somying->getWeight().'<br />';
echo 'Age : '.$somying->getAge().'<br />';
```

ชุดคำสั่งที่ 7

```
Name : Somsak
Sex : male
Height : 178
Weight : 65
Age : 28
Name : Somying
Sex : female
Height : 168
Weight : No Answer
Age : No Answer
```

ผลการทำงานของชุดคำสั่งที่ 7

จากชุดคำสั่งที่ 7 และผลการทำงานของชุดคำสั่งที่ 7 นี้จะเห็นได้ว่า ในชุดคำสั่งที่ 7 นั้นที่ `$somsak->setSex()` นั้นไม่ได้ใส่ค่าอะไรลงไปแต่ผลการทำงานนั้นก็ออกมาเป็น `male` เช่นเดิม ซึ่งเป็นชื่อตั้งต้นของพารามิเตอร์ของพฤติกรรมดังกล่าว ซึ่งช่วยให้ข้อมูลต่าง ๆ นั้นยังคงมีความถูกต้องหรือ ในบางกรณีช่วยให้โปรแกรมนั้นไม่ทำงานผิดพลาดไปจากที่ควรจะเป็นในขั้นตอน

จากบทความในตอนที่ 1 นี้เป็นเพียงการปรับพื้นความเข้าใจหลักการทำงานของการเขียนโปรแกรมเชิงวัตถุเท่านั้น ยังไม่ได้ลงลึกถึงการใช้งานทั่วไปมากมายนัก เพราะถ้าเรายังไม่เข้าใจหลักการพื้นฐานแล้วก็ยากที่จะนำไปใช้งานในระดับสูงต่อไปได้ดี โดยในตอนที่ 1 นี้ทางผู้เขียนพยายามใช้ศัพท์เทคนิคให้น้อยที่สุด แต่บางครั้งคำไทยที่ใช้อาจทำให้งุนงง ซึ่งทางผู้เขียนก็ได้ทำกำกับไว้ด้านหลังคำไทยนั้น ๆ ไว้เพื่อให้เข้าใจง่ายขึ้นสำหรับท่านที่ใช้คำเทคนิคเหล่านี้เป็นประจำครับ สำหรับตอนต่อไปนั้นผมจะแนะนำการนำเอาแม่แบบและวัตถุที่สร้างนำไปใช้งาน โดยที่เราจะยังไม่สืบทอดแม่แบบ แต่ใช้งานทั่วไปก่อนเป็นอันดับแรก

# เอกสารอ้างอิง


- **PHP Manual,** http://www.php.net, January 2007
- **Zend PHP Certification Study Guide**, Zend Technologies, August 2004
- **Concepts of Programming Languages (7th Edition),** Robert W. Sebesta, April 2005
- **The Object Model**, http://developer.apple.com/, December 2006