บทที่ 10
Self และ Parent

จากบทที่ผ่านมาได้กล่าวถึง การเรียกใช้งานพรอพเพอร์ตี้และเมธอด ประเภท Static ซึ่งในการเรียกใช้งานภายในคลาสๆ หนึ่ง แต่ในการทำงานที่ซับซ้อนนั้นจะมีการเรียกใช้ผ่านคลาสต่างๆ ที่ถูกแบ่งออกไปเป็นลักษณะคลาสแม่ (Super Class) และคลาสลูก (Sub Class) ซึ่งจะมีการเรียกใช้งานในคลาสแบ่งออกเป็น 2 ประเภท คือ self และ parent ซึ่งมีหลักการทำงานดังต่อไปนี้

10. การเรียกใช้เมธอด self และ parent

หลักการทำงานของคำสั่ง self หมายถึง การเรียกใช้งานโดยตรงที่อยู่ภายในคลาสของตนเอง ส่วนคำสั่ง parent จะใช้ในการเรียกใช้งานโดยตรงที่อยู่ในคลาสแม่ ดังตัวอย่างที่ 10.1

ตารางที่ 10.1 การเรียกใช้งาน self และ parent

```
1  <?
2  class Mam
3  {
4          protected static $Surname;
5          protected static $DNA;
6          public function __construct()
7          {
8                  echo "MAM";
9          }
10         public function MamName()
11         {
12                 echo "จันดี";
13         }
14 }
15
16 class Child extends Mam
```

```
17  {
18          public function __construct()
19          {
20                  parent::__construct();
21          }
22
23          public function show()
24          {
25                  echo "สวัสดี ";
26          }
27          public function Hello()
28          {
29                  echo self::show();
30                  echo parent::MamName();
31          }
32  }
33
34  $obj=new Child();
35  echo "<br>";
36  echo $obj->Hello();
37  ?>
```

ผลลัพธ์

```
MAM
สวัสดี  จันดี
```

จากโปรแกรมในตัวอย่างที่ 10.1 สามารถอธิบายหลักการทำงานของโปรแกรมได้ดังต่อไปนี้ คือ เมื่อมีการสร้างออบเจ็กต์หรืออินสแตนด์จากคลาส Child (บรรทัดที่ 36) จะมีการสร้างคอนสตรัคเตอร์โดยอัตโนมัติ ซึ่งในคลาส Child ได้มีการสร้างคอนสตรัคเตอร์ไว้ (บรรทัดที่ 20) ซึ่งภายในคอนสตรัคเตอร์นี้จะมีคำสั่ง parent::__construc() ซึ่งหมายถึงเรียกใช้คอนตรัคเตอร์ของคลาสแม่ (คลาส Mam) ซึ่งเมื่อไปดูที่คอนสตรัคเตอร์ของคลาส Mam (บรรทัดที่ 6) จะมีคำสั่งให้แสดงข้อความว่า “Mam” (นั่นก็คือผลลัพธ์ในบรรทัดที่ 1)

ในการประมวลผลต่อมา (บรรทัดที่ 37) นั่นก็คือ ให้ทำการปัดบรรทัด 1 บรรทัด (คำสั่ง
 คือ คำสั่งให้ปัดบรรทัด) จะสังเกตได้ว่าผลลัพธ์จะอยู่บรรทัดต่อมาหลักการประมวลผล

หลังจากประมวลผลในบรรทัดที่ผ่านมาดังกล่าว มีการเรียกใช้ออบเจ็กต์ไปยังเมธอด Hello (บรรทัดที่ 38) ในเมธอด Hello จะทำการเรียกใช้เมธอดแบบ self::show() นั่นก็คือให้เรียกใช้เมธอด show ที่อยู่ภายในคลาสของตนเอง (เมื่อมีการเรียกใช้เมธอดภายในคลาสตนเองต้องใช้คำสั่ง self เท่านั้น) ซึ่งเมธอด show จะให้แสดงข้อความว่า “สวัสดี” (ดังในผลลัพธ์ในบรรทัดที่ 2) หลังจากนั้นมีการเรียกใช้เมธอดแบบ parent:: MamName() ในคลาสแม่ นั้นก็คือคลาส Mam ในเมธอด MamName ซึ่งเมธอดนี้ก็จะให้แสดงข้อความว่า “จันดี” (ดังในผลลัพธ์ของโปรแกรม)

10.2 การเรียกใช้พรอพเพอร์ตี้ self และ parent

ในการเรียกใช้งานพรอพเพอร์ตี้แบบ self และ parent หลักการคล้ายกับการใช้งานเมธอด ดังตัวอย่างที่ 10.2

ตัวอย่างที่ 10.2 การเรียกใช้เมธอดแบบ static

```
1   <?
2   class Mam
3   {
4           protected static $Surname;
5
6   }
7
8   class Child extends Mam
9   {
10          protected static $Surname;
11          public function Hello()
12          {
13                  self::$Surname="จันดี";
14                  parent::$Surname="สมใจ";
15                  echo self::$Surname." ".parent::$Surname;
16          }
17  }
18
```

```
19  $obj=new Child();
20  $obj->Hello();
21  ?>
```

ผลลัพธ์ของโปรแกรม

```
จันดี สมใจ
```

หลังจากที่มีการสร้างออบเจ็กต์ และมีการเรียกใช้เมธอด Hello (บรรทัดที่ 19-20) เมธอดที่ถูกเรียกใช้ จะมีการเรียกใช้คุณสมบัติตัวที่หนึ่ง นั่นคือ self::$Surname="จันดี" หมายถึงการเรียกพรอพเพอร์ตี้ Surname ในคลาส Child มาใช้งานโดยให้มีค่าเท่ากับ จันดี และตัวที่สอง parent::$Surname="สมใจ" หมายถึง การเรียกใช้พรอพเพอร์ตี้ Surname ที่อยู่ในคลาส Mam มาใช้โดยกำหนดให้มีค่าเท่ากับ สมใจ และให้แสดงค่าทั้งสอง (บรรทัดที่15) ซึ่งได้ผลลัพธ์ดังตัวอย่าง

จากการทำงานดังกล่าวสังเกตได้ว่าค่าพรอพเพอร์ตี้มีชื่อเดียวกันคือ Surname แต่ไม่ใช่พรอพเพอร์ตี้เดียวกัน เพราะอยู่คนละคลาสดังเห็นได้ว่าผลลัพธ์ค่าที่ได้ ซึ่งหลักการตรงนี้ก็ใช้วิธีการเรียกใช้งานผ่าน self และ parent