Late Static Binding

Let’s start with a simple example.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | <?php    classModel{  protected static $tableName='Model';  public static function getTableName(){  return self::$tableName;  }  }    Class User extends Model{  Protected static $tableName='User';  }    Echo User::getTableName();// Model, not User |

How it works.

* First, we created a Model class that has $tableName static property with value Model and a getTableName() static method that returns the value of the $tableName.  Notice that we used the self and the operator :: to access static property inside the Model class.
* Second, we created another class named User that extends the Model class. The User class also has $tableName static attribute.
* Third, we called the getTableName() method of the User class. However, it returns Model instead of User. The reason is that self is always resolved to the class in which the method belongs. It means that if you define a method in a parent class and call it from a subclass, the self does not reference to the subclass as we expect.

To overcome this issue, as of version 5.3, PHP introduced a new feature called **PHP static late binding**. Basically, instead of using the self, you use the static keyword that references to the exact class that was called at runtime.

Let’s modify our example above:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | <?php    Class Model{  Protected static $tableName='Model';  Public static function getTableName(){  return static::$tableName;  }  }    Class User extends Model{  Protected static $tableName='User';  }    echo User::getTableName();// User |

Now we get the expected result.

Notice that the static:: can only refer to [static properties and static methods](http://www.zentut.com/php-tutorial/php-static/).