Example 3-6. Basic Dog class with return statement—dog.php

```
<?php
class Dog
{
private $dog_weight = 0;
private $dog_breed = "no breed";
private $dog_color = "no color";
private $dog_name = "no name";

function get_properties()
{

return "$this->dog_weight,$this->dog_breed,$this->dog_color.";

}
}
?>
```

■ **Note** In PHP 7, you can enable Scalar Type Hints. PHP 7 provides the developer the ability to declare the data type that is expected to be returned. The function in Example 3-6 could be coded as follows.

```
declare(strict_types=1);
function get_properties() : string
{
return "$this->dog_weight,$this->dog_breed,$this->dog_color.";
}
```

If the `declare` statement is not included or `strict_types=0`, the data type will not be enforced. The current valid data types that can be used are `string`, `int`, `float`, and `bool`.

Since Scalar Type Hints are not backward compatable, they are not used in the examples in this book.

You now need to adjust the `lab.php` file to be able to accept what has been passed back from the `get_properties` method (`display_properties` is renamed `get_properties` to reflect that it no longer displays the properties; it now returns them). You can accomplish this by creating a property in the `lab.php` file to receive what has been passed back from the `get_properties` method.

```
$dog_properties = $lab->get_properties();
```

If you were to use the `print` function to display `$dog_properties` at this point, you would display:

```
no weight, no breed, no color
```

However, we intended to produce a similar result as was shown previously. You can do this, but you need to be able to break the string into three pieces based on the "," delimiter. Luckily there are PHP methods available that can easily accomplish this task. The `explode` method will break a string based on a delimiter.