Example 3-8. The lab.php file with set methods and error checking

```
<?php
require_once("dog.php");
$lab = new Dog;
// -------------------Set Properties-------------------------
$dog_error_message = $lab->set_dog_name('Fred');
print $dog_error_message == TRUE ? 'Name update successful<br/>' : 'Name update not
successful<br/>';

$dog_error_message = $lab->set_dog_weight(50);
print $dog_error_message == TRUE ? 'Weight update successful<br />' : 'Weight update not
successful<br />';

$dog_error_message = $lab->set_dog_breed('Lab');
print $dog_error_message == TRUE ? 'Breed update successful<br />' : 'Breed update not
successful<br />';

$dog_error_message = $lab->set_dog_color('Yellow');
print $dog_error_message == TRUE ? 'Color update successful<br />' : 'Color update not
successful<br />';
//----------------------------Get Properties--------------------------
$dog_properties = $lab->get_properties();
list($dog_weight, $dog_breed, $dog_color) = explode(',', $dog_properties);
print "Dog weight is $dog_weight. Dog breed is $dog_breed. Dog color is $dog_color.";
?>
```

In Example 3-8, `lab.php` now has the ability to pass information into the properties in the `$lab` object of the `Dog` class. It also determines if the update for each property was successful and responds accordingly. There are opportunities in this example to be more efficient with the amount of code that you have created. However, we will hold off on efficiency until you have gathered a few more skills.

The `lab.php` code now calls a `set` method for each property to be updated (`set_dog_name`, `set_dog_breed`, `set_dog_weight`, and `set_dog_color`) and passes information into each method. Notice that strings are passed into each method, except for the `set_dog_weight` method, which accepts an integer (whole number) value.

You now need to create `set` methods within the `Dog` class. Each method now accepts a parameter (string or integer) and returns a '`TRUE`' or '`FALSE`' value. The method is created in a style similar to the previous `get_properties` method that you created. Let's keep the validation process simple for now and you'll learn how to improve it in later chapters.

```
function set_dog_name($value)
{
$error_message = TRUE;
(ctype_alpha($value) && strlen($value) < 21) ? $this->dog_name = $value : $error_message = FALSE;
return $error_message;
}
```

The `set_dog_name` method will accept a string into the `$value` property (parameter) defined in the function header (`function set_dog_name($value)`). Next, the method creates a property `$error_message` and provides an initial value of `TRUE`. This property (along with the `$value` property) will only exist while the method is executing. As soon as the execution hits the } closing bracket, these properties will no longer be available.