When an instance of the dog class is created by `create_object`, dog is again passed in to indicate that an instance of the dog class is needed.

```
$container->set_app("dog");
$dog_app = $container->get_dog_application("dog");
$method_array = get_class_methods($lab);
$last_position = count($method_array) - 1;
$method_name = $method_array[$last_position];
$returned_array = $lab->$method_name("ALL");
```

The same code that was used to call the `delete`, `insert`, and `update` methods in `dog_data` is used to call the display method. However, `ALL` is passed in instead of the properties array. `ALL` tells the `dog_data` display method to return all the dog records. All the records that are returned by `dog_data` are dropped into `$return_array`.

Now that the `dog_interface` has all the records (in `$return_array`), it can format the code to display the dogs list box. The code used is similar to the code to create the `breeds` list box.

```
$resultstring = "<select name='dogs' id='dogs'>";
$resultstring = $resultstring . "<option value='-1' selected>NEW</option>";
```

The property `$resultstring` will hold all the code for the list box. First the HTML select tag is created with an ID of `'dogs'`. Then the first row of the list box is created for the user to select NEW if they want to insert a new dog. Notice that the value for NEW is `-1`. That's also the default value for the hidden property `'index'` on the HTML form in `lab.php`. The code you looked at early in this chapter will determine that `-1` is an indication to fill the HTML form objects with the default settings for the dog information. This is true either if the user selects NEW or does not select anything in the dogs list box.

```
foreach ($returned_array as $column => $column_value)
    {
        $resultstring = $resultstring . "<option value='$column'>" . $column_value['dog_name'];
        $resultstring .= “&nbsp;&nbsp;&nbsp;" . $column_value['dog_breed'] . "</option>";
    }
$resultstring = $resultstring . "</select>";
```

A `foreach` loop will loop through the dogs array (contained in `$returned_array`) and build the remaining rows of the list box using the `dog_name` and `dog_breed` from each of the dog entries in the array. After all dogs have been placed in the list box, the list box is closed using the HTML `</select>` tag.

```
print $result . "|" . $resultstring . "|" . '{ "dogs" : ' . json_encode($returned_array) . "}";
```

`lab.php` expects to receive the `breeds` list box code in the first position, the `dogs` list box code in the second position, and the complete dogs array in the third position. `$result` already contains the complete breeds list box. `$resultstring` contains the new dogs list box. `$return_array` still contains all the dog records. `lab.php` expects the outer array of the dogs array to be labeled as `'dogs'`. The display method from `dog_data` does not pass the outer array back. The previous code will create a dogs array that holds all the individual dog arrays in it. Notice that the `$return_array` has been converted to JSON code when returned. The JavaScript code in `lab.php` will verify that it is properly formatted JSON code when it is received.

Let's look at the complete `dog_interface` program.