The object's location in memory is returned in a similar way as the array's location was passed into the constructor in the previous example. You can think of it as the new object is temporarily "contained" within the dog_container object. However, the object is returned (to dog_interface).

> *Programming note—What? What really happens is that the address in memory of the $dog_object is passed to the calling program (dog_interface). This allows the calling program to have access to the object, along with the $dog_container object. Thus, there is only one copy of the $dog_object in memory but two different program blocks can use it. If one of the blocks (dog_container or dog_interface) closes, the other object still has access to it, until it also closes. Then the garbage collector will remove the $dog_object from memory.*

Example 4-12. The get_breeds class

```
<?php
class GetBreeds {
function __construct($properties_array)
{ //get_breeds constructor
if (!(method_exists('dog_container', 'create_object')))
{ exit;}}
private $result = "??";
public function get_select($dog_app)
{ if (($dog_app != FALSE) && ( file_exists($dog_app))) {
     $breed_file = simplexml:load_file($dog_app);
     $xmlText = $breed_file->asXML();
     $this->result = "<select name='dog_breed' id='dog_breed'>";
     $this->result = $this->result . "<option value='-1' selected>Select a dog breed</option>";
    foreach ($breed_file->children() as $name => $value)
    {    $this->result = $this->result . "<option value='$value'>$value</option>";  }
      $this->result = $this->result . "</select>";
              return $this->result;
    } else {
              return FALSE;
    }
}
}
?>
```

As you can see from Example 4-12, only minor changes were needed. As mentioned, a class was declared and a constructor was added. The constructor verifies that this class was created from a program that contains both the dog_container and create_breed_app methods. This security attempts to keep other programs from knowing that the file names and locations for the Dog application that reside in the dog_application.xml file.

Example 4-13. The dog_interface.php file

```
<?php
function clean_input($value) {
$bad_chars = array( "{", "}", "(", ")", ";", ":", "<", ">", "/", "$" );
$value = str_ireplace($bad_chars,"",$value);
 $value = htmlentities($value);
$value = strip_tags($value);
```