You will look at the PHP changes necessary in later sections of this chapter. For now, let's assume that `dog_interface` (interface tier) will return this information, which was retrieved from `dog_data` (data tier) via the dog methods (business rules tier). You will also assume that similar coding to the `dog_breeds` program (Chapter 4) will be created to produce a dogs list box.

In addition, `lab.php` needs access to all the information from a particular dog that has been selected. The code will need to place all the information in the `dog_name` and `dog_weight` text boxes, the `dog_breeds` list box, and the `dog_color` radio buttons. You could accomplish this task in two ways. One is to allow the users to select the dog from the list box and then recall the `dog_interface` program to request the particular dog from the dog class, which in turn will request the information from the `dog_data` class. However, this is requiring an additional, unnecessary, call across the Internet to request the information. Instead, you can gather all the necessary information (`dog_breed` list box information, `dogs` list box information, and complete information of all dogs in the current data storage) when the user first calls the `lab.php` interface. You can use the current JavaScript AJAX code (Chapter 4) with just a couple of changes to retrieve all the necessary information.

You must make sure that the user indicates what type of operation is requested (at least if it is an insert or change/delete) before you populate the form objects (text boxes, list box, and radio button). You can accomplish this by not displaying the form objects until a selection has been made. You can make a slight adjustment to the combination of JavaScript code and CSS code, used in Chapter 4, to require the users to select from a dogs list box before the form is displayed.

■ **Note** The process you are about to look at is a very common practice used in web applications. Many web applications return data in an array, JSON, or XML format. The interface (in this example, `lab.php`) then can use JavaScript to retrieve the information needed.

Programming note—Shopping carts use a similar technique. As the customers select items to purchase, the items are placed in a data object (probably a JSON object) on the client machine. When the customer begins the process of checking out, the data is transferred to the server. This allows the customer to make changes that will not cause additional calls to the server. Since the purchase information is not considered a security risk, this is usually a safe procedure. Of course, this would not be a good way to handle credit card information.

Using the format shown in Figure 8-1, the user is forced to select `NEW` or a dog from list box before proceeding. Once the user has made a choice, the HTML form can be displayed with the default values (for an insert) or the current values for the dog selected. CSS code will initially turn the display of the buttons (and the form) to `"none"`. JavaScript code (which you will look at soon) will change the display of the correct button(s) and the form to `"inline"` to display at the proper time.