|  |  |
| --- | --- |
| Course Name | ITD 3243 – Server Side Programming |
| Instructor | Mark L. Pranger |
| Student Name | Derrick McFall |
| Due date | 30Jan19 |
| Grade | <grade earned here> |
| Grading Comments | <instructor comments here> |

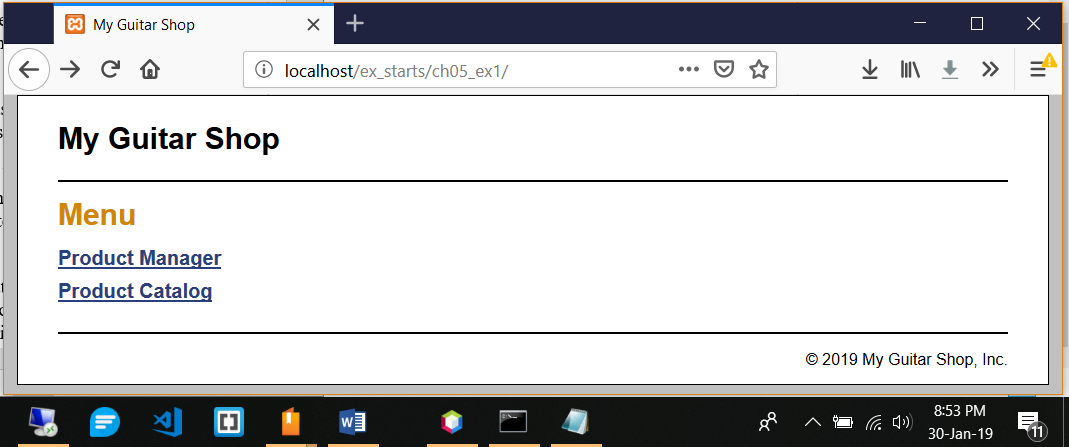
Module 4

Exercise 5-1

This exercise has you enhance the Guitar Shop application. That will give you a chance to use some of the skills that were presented in this chapter.

Test the Guitar Shop application

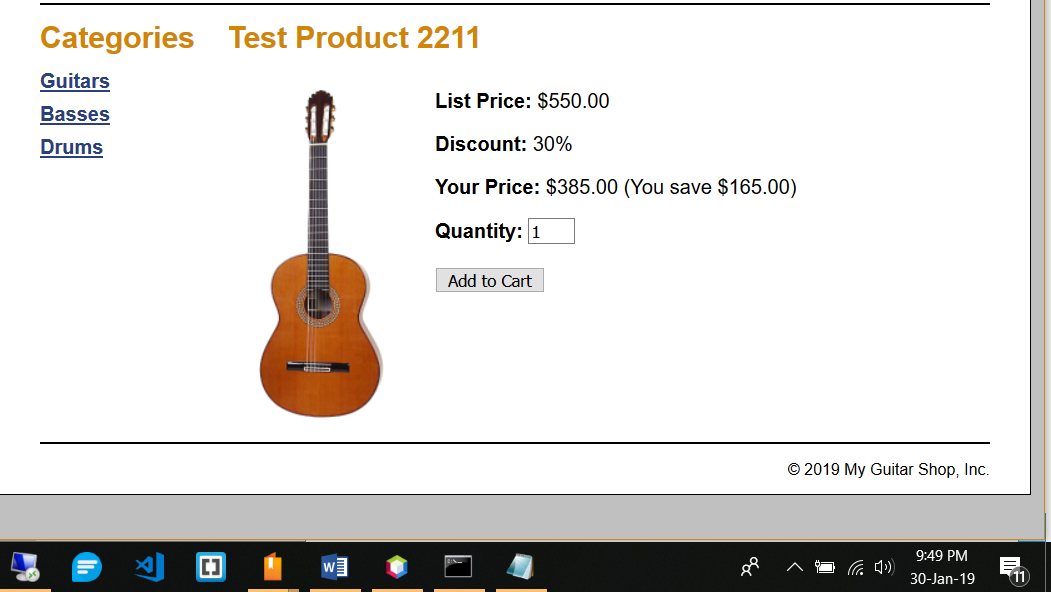
1. Start the Chrome browser and run the application in the ex\_starts/ch05\_ex1 directory. This should display a menu that lets you navigate to the Product Manager application or the Product Catalog application.



2. Use the Product Manager application to add a new product to the database with Guitars as the category, test1 as the code, Test Product 2211 as the name, and 550.00 as the list price.



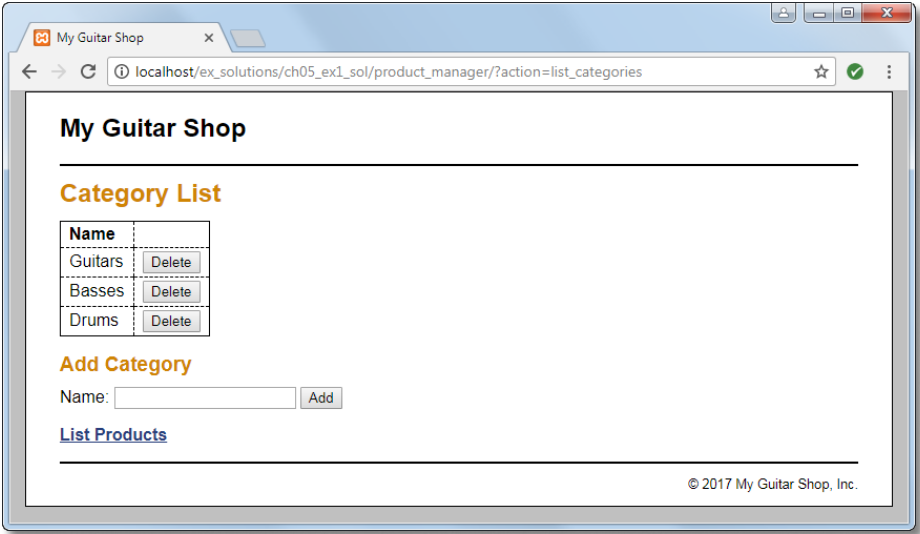
3. Go to the Product Catalog application. To do that, you can use the Back button or run the application again. Then, view the product that you just added. Note that it doesn’t display an image for the product. To fix that, go to the images directory and change the name of the test.png file to test1.png.

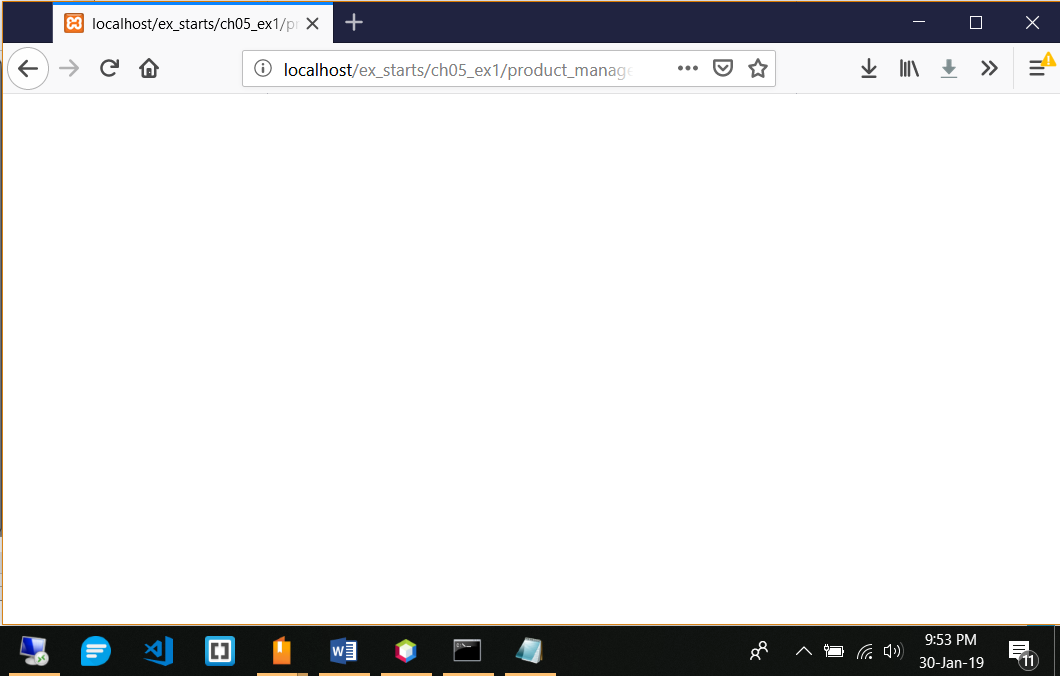


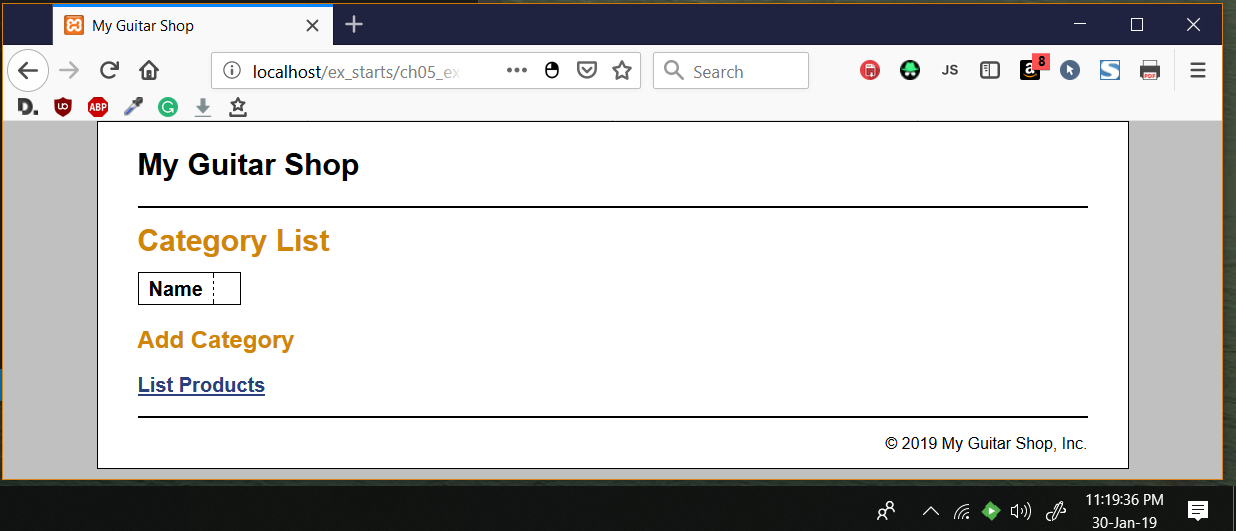
4. Go to the Product Manager application again. Then, click on the List Categories link at the bottom of the page. Note that this link doesn’t display a page, even though it is coded correctly. You’ll fix this later when you enhance the index.php page for this application. Now, click the Back button.

**Enhance the Product Manager application**

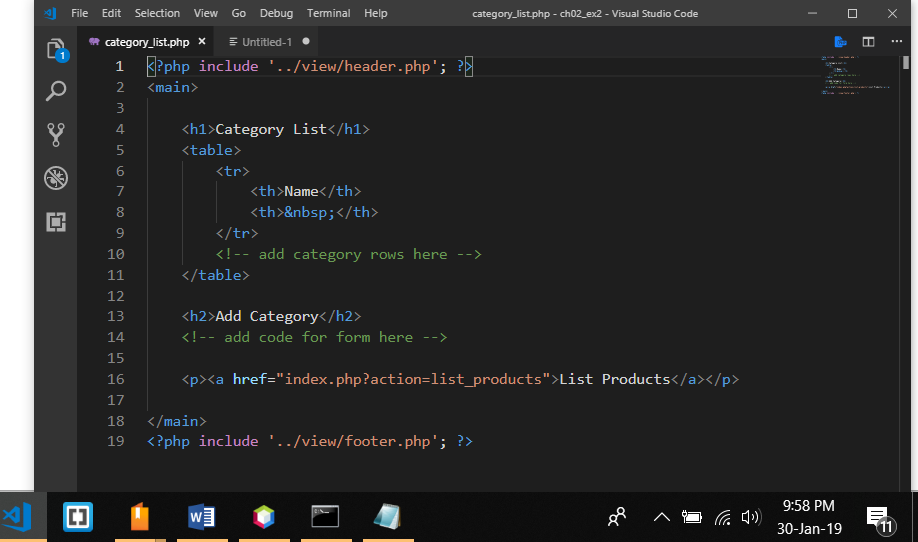
Now, you’ll add a page to the Product Manager application that lets you add or delete categories. This is similar to what you did in exercise 4-1, but using the MVC pattern. The new page should look like this:



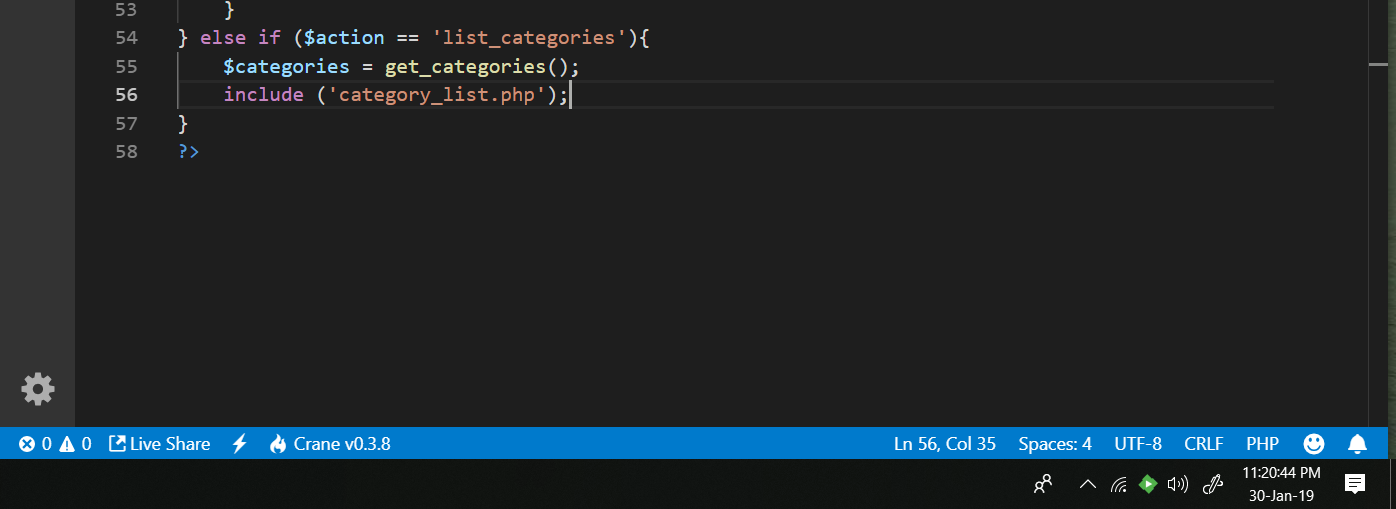


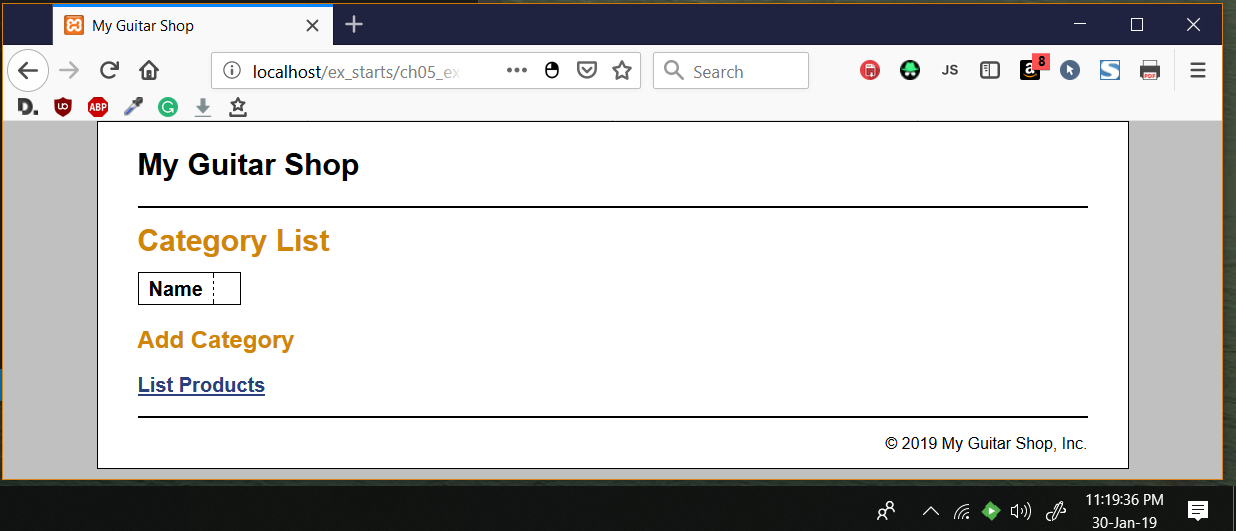


5. Open the category\_list.php file in the product\_manager directory. It contains some of the headings that you’ll need for this page, and a link back to the Product List page.

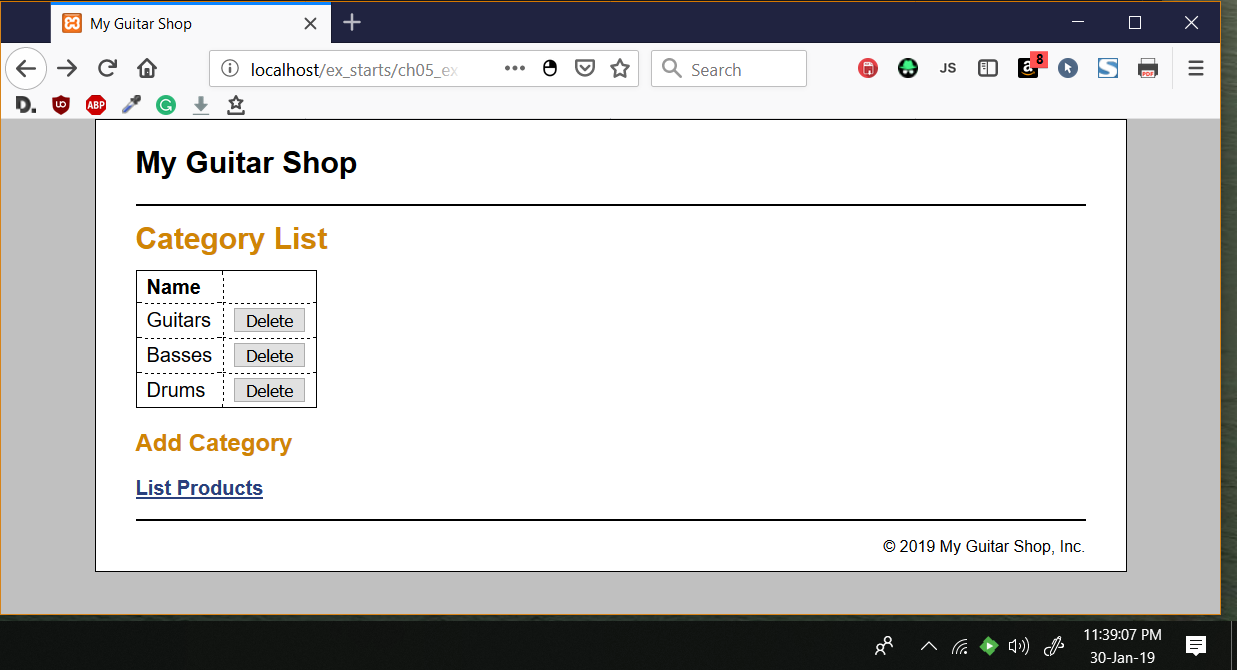


6. Open the index.php file in the product\_manager directory and add an action that displays the Category List page. Then, test this to make sure it works correctly. At this point, the page should only display some of the headings.

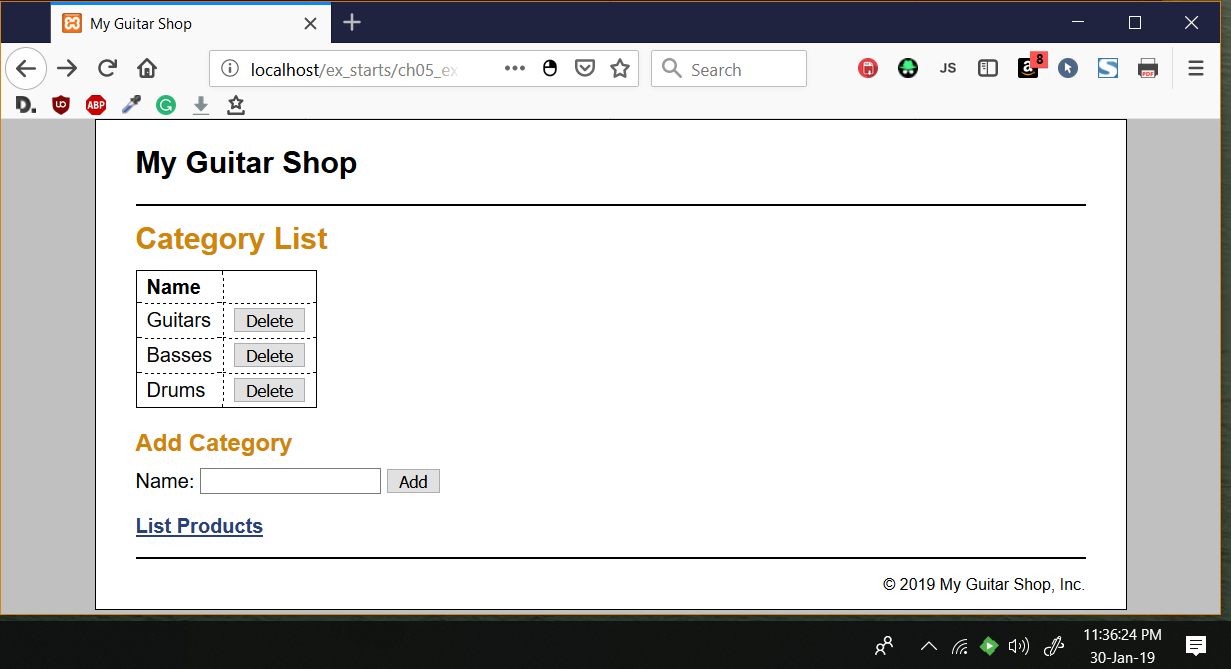




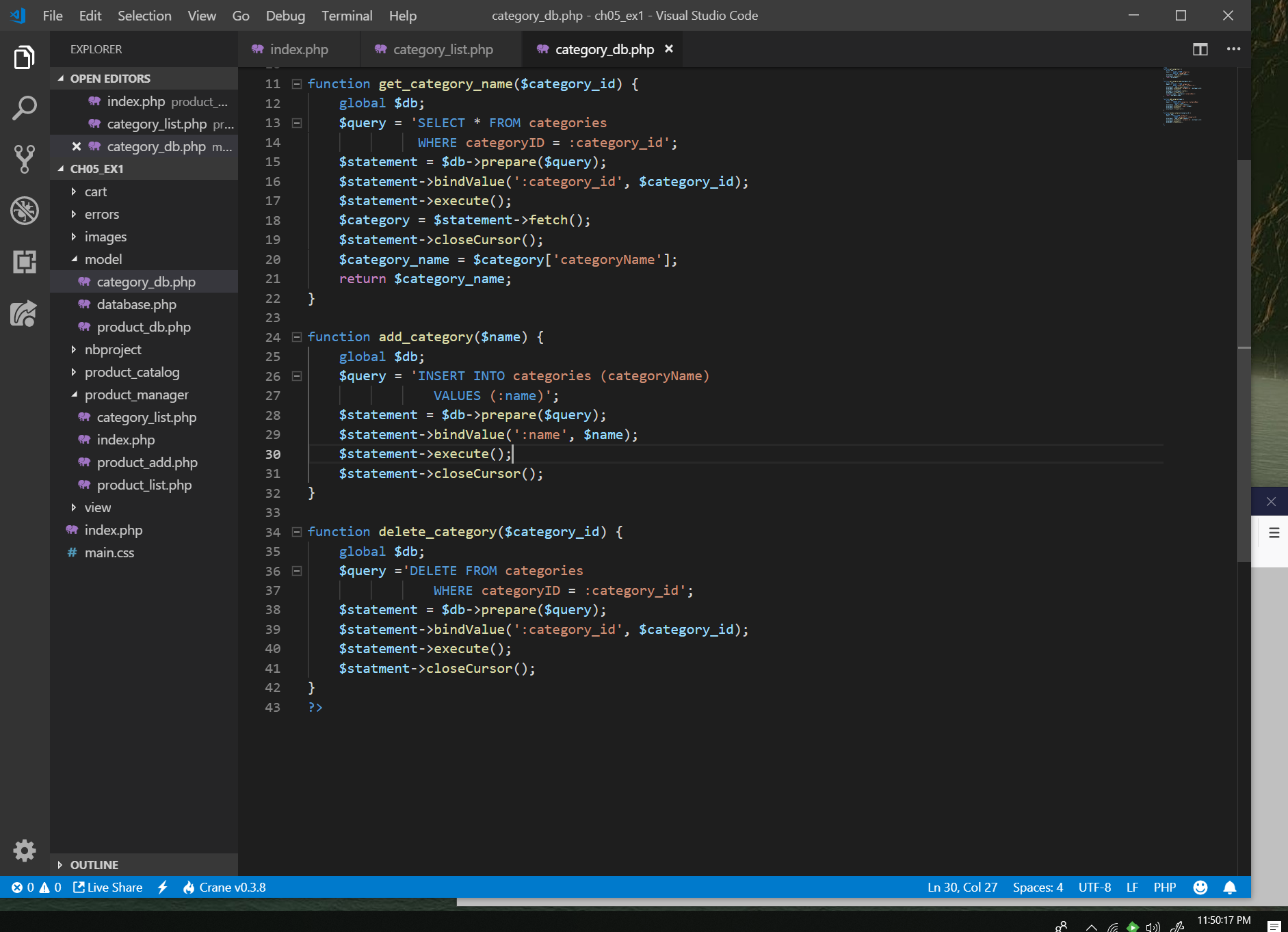
7. In the category\_list.php file, write the code that creates the category table shown above with all of the category names in the first column and Delete buttons in the second column. Then, test this to make sure the page displays the table correctly.



8. In the category\_list.php file, write the code that lets the user add a category to the database. This code should consist of a form that accepts the name for a new category followed by a Submit button that displays “Add”.



9. Open the model/category\_db.php file and add two functions to it that add and delete categories.



10. Open the index.php file in the product\_manager directory and add two new actions to it. The first action should add a category to the database. And the second action should delete a category from the database.

Tip: To return to the Category List page after adding or deleting a category, you can pass an action to the controller with a statement like this:

header('Location: .?action=list\_categories');

11. Test the application by adding two categories. Then, navigate to the Product List page and note that the list of categories includes the new categories. Next, navigate to the Add Product page and note that the drop-down list includes the new categories.

12. Test the application by deleting the categories that you just added. However, don’t delete any of the existing categories because that will lead to products without categories. If necessary, though, you can restore the database by running the create\_db.sql script again as described in the appendixes.

13. If the formatting of your page isn’t exactly like the one above, don’t worry about that. The focus here is on web programming, not HTML and CSS.

Refactor the Product Catalog application

14. Open the product\_list.php and product\_view.php files in the product\_catalog directory. Note that these files use the same code in the <nav> tag to display the list of navigation links for each category.

15. Create a file named categories\_nav.php in the view directory, and copy all the code for the <nav> tag from the product\_list.php file to the category\_nav.php file. Then, replace the code for the <nav> tag in the product\_list.php and product\_view.php files with the appropriate include statements.

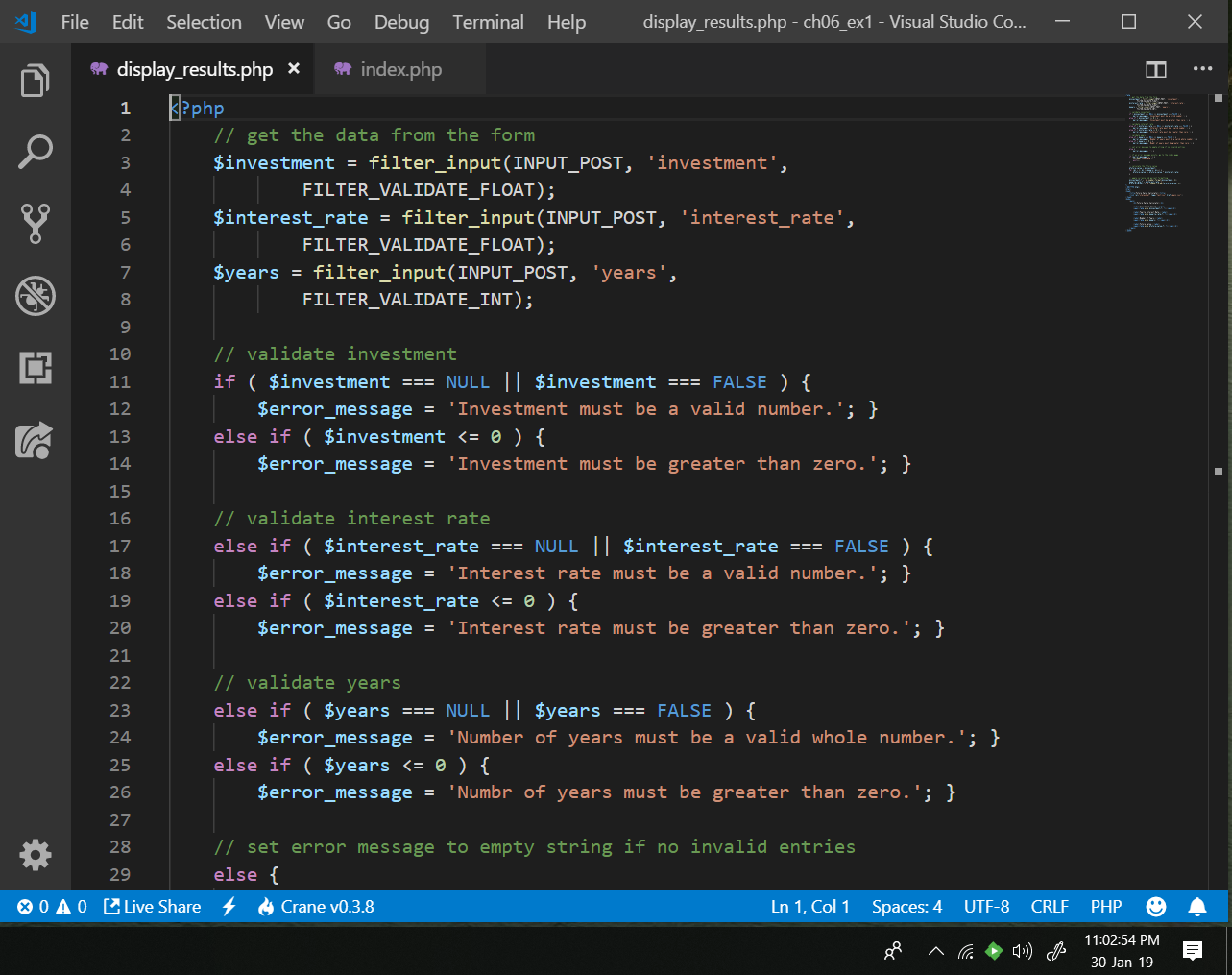
16. Test these changes to make sure your refactoring works.

Exercise 6-1

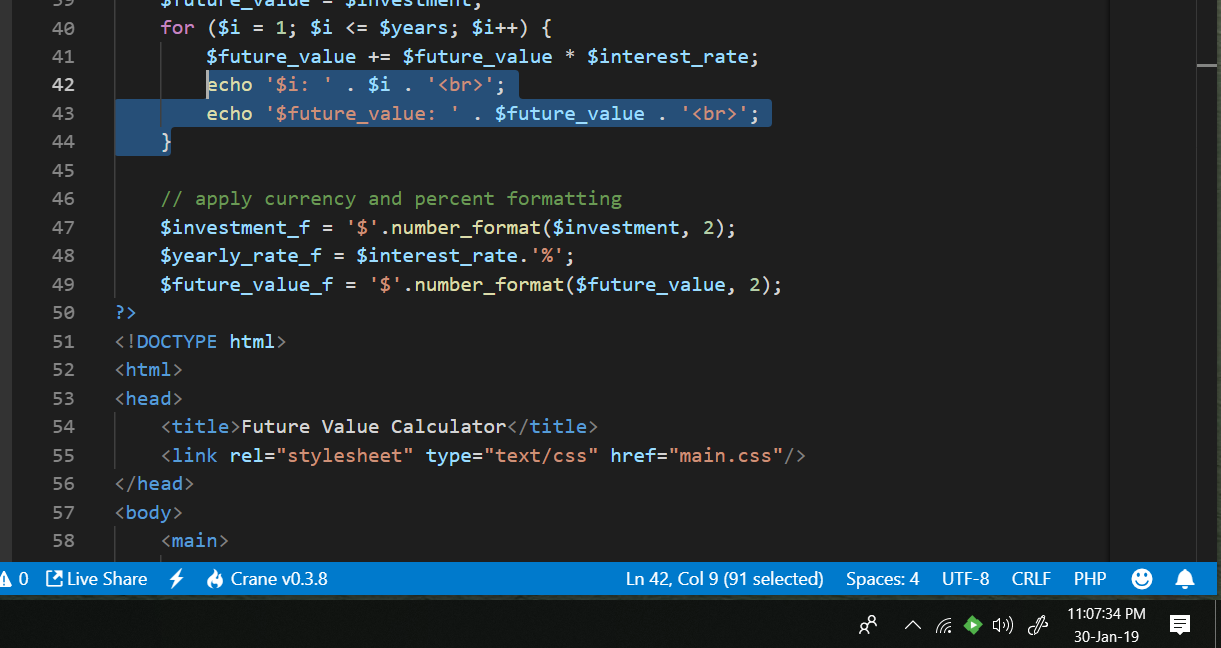
Exercise 6-1 Trace with echo statements

In this exercise, you’ll use echo statements to trace the execution of the Future Value application.

1. Open the display\_results.php file for the Future Value application that’s stored in the ex\_starts\ch06\_ex1 directory.



2. Add echo statements like those in figure 6-3 to trace the execution of the code. Then, run the application to see how the echo statements work.



Exercise 6-2

1. Start NetBeans and open the ch06\_ex2 application in the ex\_starts directory. Then, open the display\_results.php file in the editor.

2. Set a breakpoint in the display\_results.php file on the statement before the for loop that calculates the future value.

3. Right-click on the project, select the Debug command to begin debugging. If this displays a dialog box warning that a firewall has blocked some features of NetBeans, allow NetBeans to communicate on private and/or domain networks.

4. Switch back to NetBeans after the browser opens. The debugger should stop the application at the first PHP statement it encounters. Then, click on the Step Into button repeatedly to step through the code until the first web page is displayed.

5. Switch to the browser to view the first page. Then, enter valid values, click on the Calculate button, and switch back to NetBeans. The application should stop at the next PHP statement it encounters.

6. Click on the Step Into button to execute three statements. Then, click on the Variables tab to display the current variables.

7. Click on the plus sign of the Superglobals variable in the Variables tab, and click on the plus sign for the \_POST array. That will display the values of the three user entries that were passed to the server.

8. Click on the Continue button to continue to the breakpoint that you set. Then, use the Step Into button to execute the loop. Note how the variables in the Variables tab change as you execute the loop.

9. Click on the Continue button to finish the calculation and display the next web page. To view this page, switch to your browser.

10. Run the application again and experiment with the Step Into, Step Over, and Step Out buttons as you step through the code of the application. At each step, notice the values that are displayed in the Variables tab.

11. When you’re through experimenting, remove the breakpoint and click on the Finish Debugger Session button. Close the browser and then close the project.

Exercise 6-3

1. Start NetBeans, and open the ch06\_ex3 application that’s in the ex\_starts directory.

2. Open the index.php file in the product\_manager directory. Then, set a break point on the first line of the if statement.

3. Run the debugger and step through the application. Switch back and forth between NetBeans and your browser as necessary.

4. Display the Variables window and note how the variables are displayed and how they change as you step through the application.

5. Use the Step Into button and note how NetBeans opens the source code files in the text editor when necessary.

6. Use the Continue button and note how it displays the browser or skips to the breakpoint that you set in step 2 depending on where you are in the application.

7. When you’re through experimenting, end the debugging session, remove the breakpoint, and close the project.