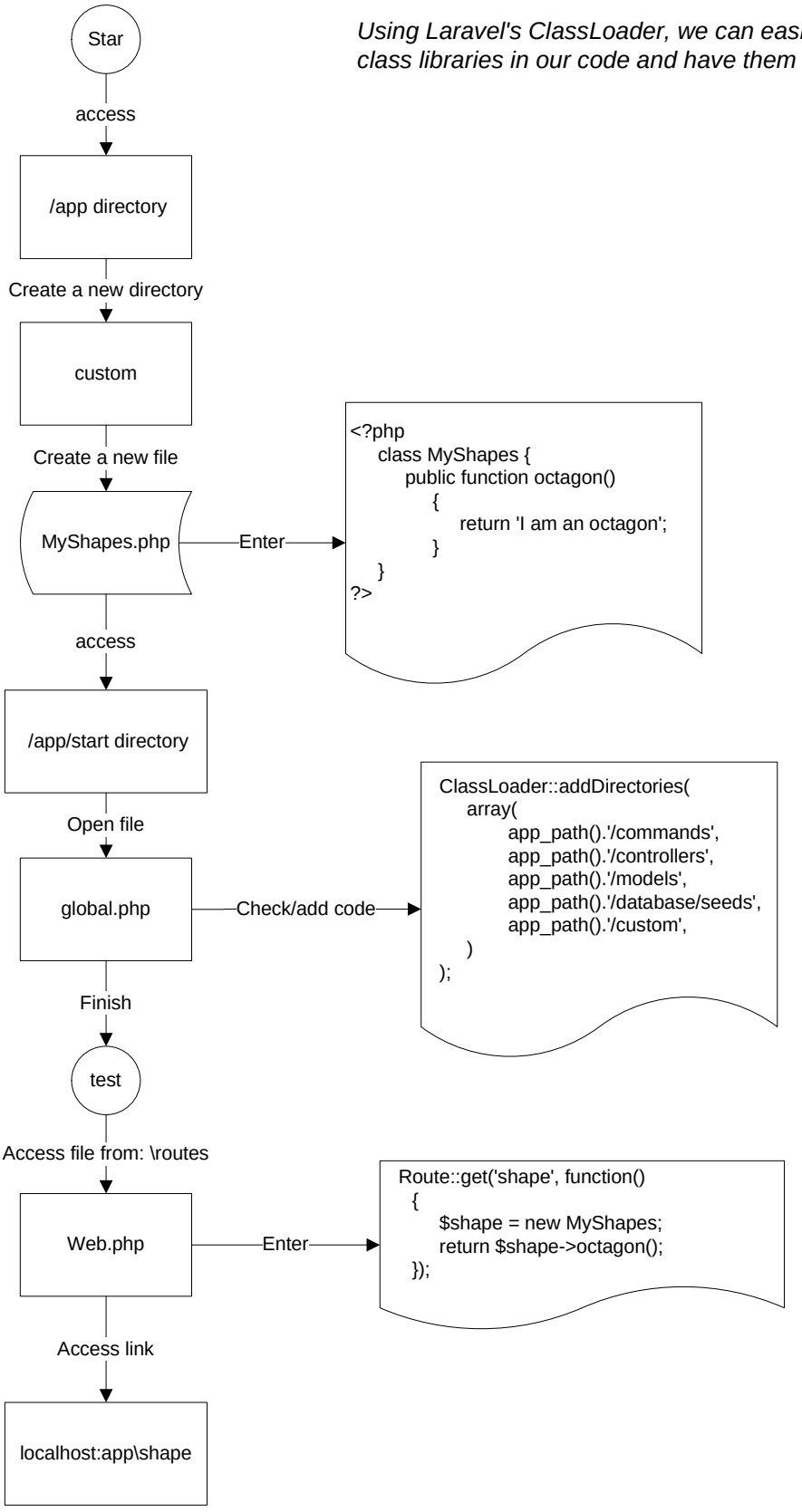
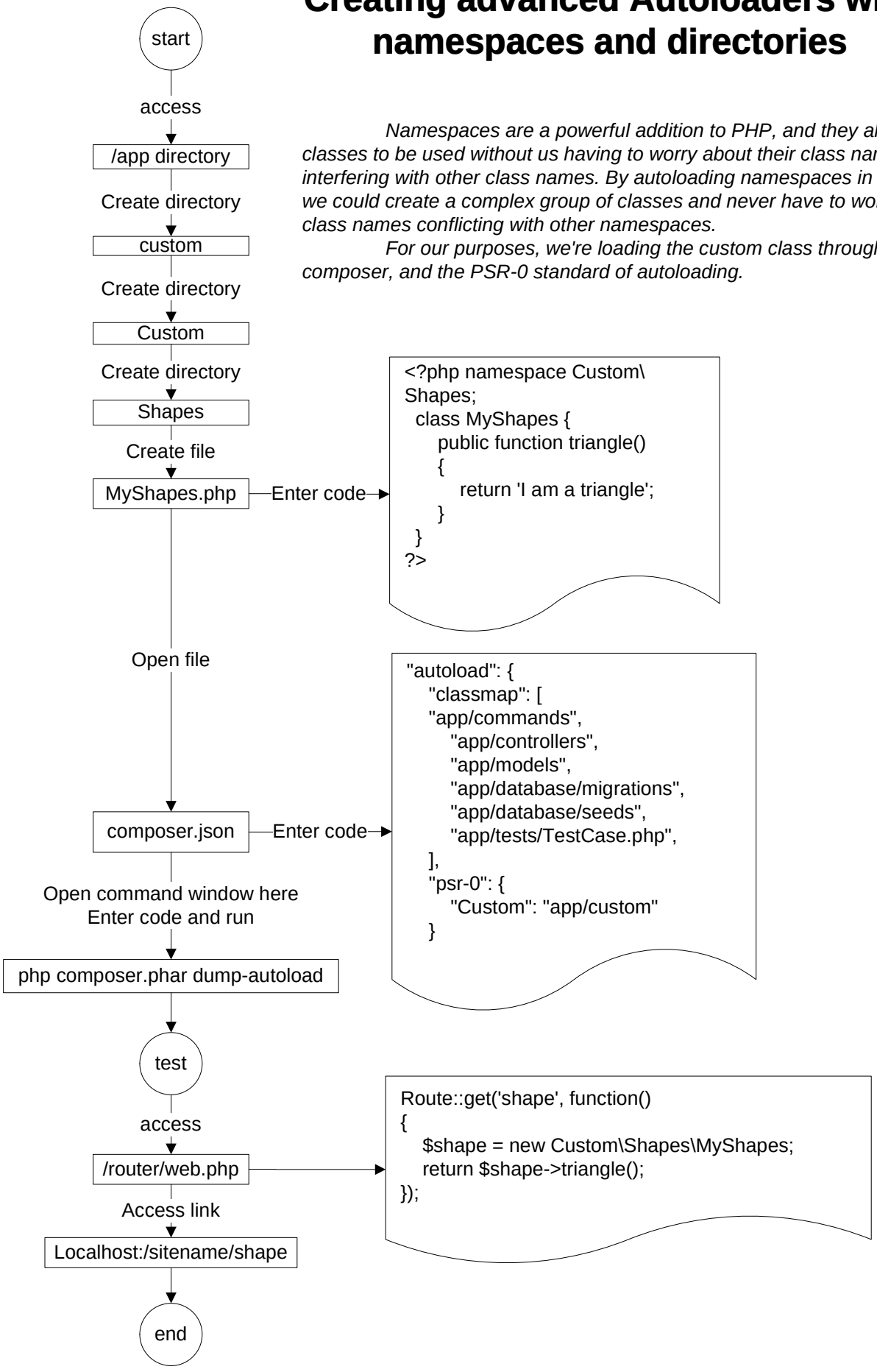
Using Autoloader to map a class name to its file

Using Laravel's ClassLoader, we can easily include any of our custom class libraries in our code and have them readily available



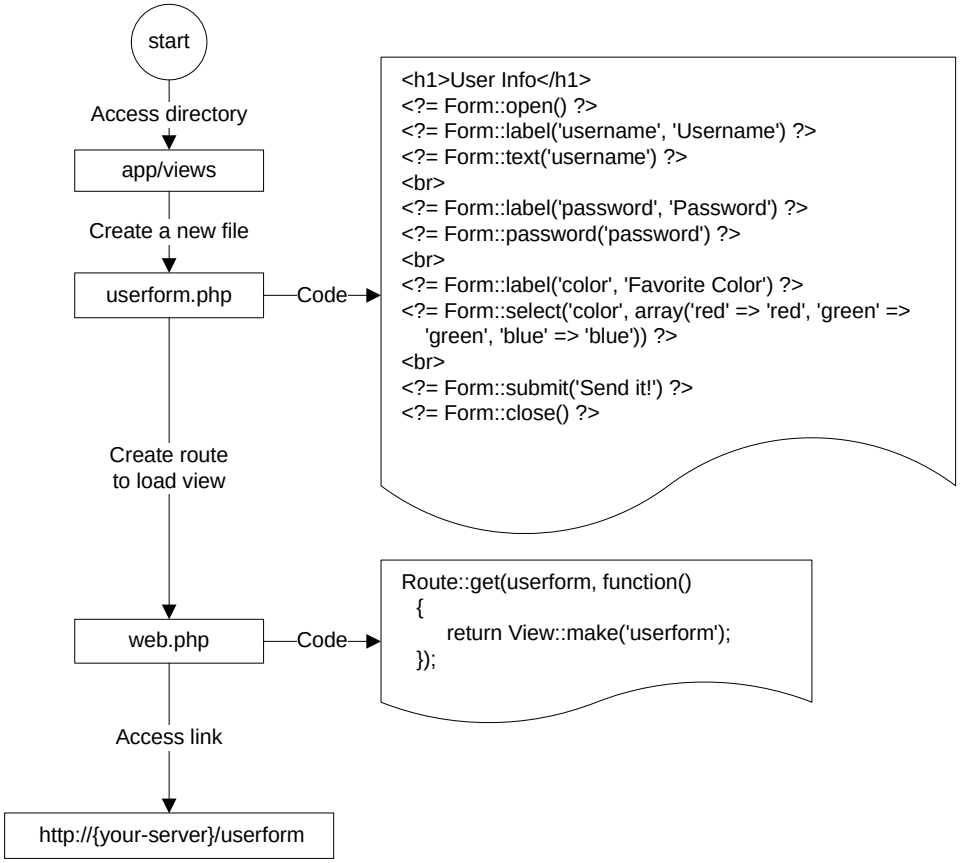
Creating advanced Autoloaders with namespaces and directories



Namespaces are a powerful addition to PHP, and they allow our classes to be used without us having to worry about their class names interfering with other class names. By autoloading namespaces in Laravel, we could create a complex group of classes and never have to worry about class names conflicting with other namespaces.

For our purposes, we're loading the custom class through composer, and the PSR-0 standard of autoloading.

Creating a simple form



For this task, we created a simple form using Laravel's built-in Form class. This allows us to easily create form elements with minimal code, and it's W3C (World Wide Web Consortium) compliant.

First, we open the form. Laravel automatically creates the <form> html, including the action, the method, and the accept-charset parameters. When there are no options passed in, the default action is the current URL, the default method is POST, and the charset is taken from the application config file.

Next we create normal text and password input fields, along with their labels. The first parameter in the label is the name of the text field and the second is the actual text to print. In the form builder, the label should appear before the actual form input.

The form select requires a second parameter, an array of the value in the drop-down box. In this example, we're creating an array using the 'key' => 'value' syntax. If we want to create option groups, we just need to create nested arrays.

Finally, we create our Submit button and close the form.

Most of Laravel's form methods can also include parameters for a default value and custom attributes (classes, IDs, and so on). We could also use Form::input() for many fields, if we didn't want to use the specific methods. For example, we could have Form::input('submit', NULL, 'Send it!') to create a submit button.