Web Programming in PHP with Design Patterns

by 1098399

Writing Competency Paper

October 4, 2004

Abstract:

Programmers face many problems when designing and programming a web site. Most of these problems have been dealt with before and by industry professionals with years of programming experience behind them. It is possible for a programmer to make use of this experience by way of design patterns, instead of coming up with solutions to problems on their own from scratch. This paper wishes to look at some of the problems faced by web programmers today and how design patterns can help in the web site creation process, specifically when the programmer is using PHP. Some experience with programming and object-orientated design is assumed.

Web Programming in PHP with Design Patterns

1. Introduction:

Over the years, as Internet web sites have become more and more complex, the need for good and careful design has become a major issue in the web development community. Although the issue of design has been dealt with in classical software engineering in great detail, it has only begun to be dealt with by web developers. The goal of this paper is to look at web site design from the perspective of PHP programming and see how using generalized solutions to problems, Design Patterns, can greatly aid in the design process.

2. The Programming Language:

PHP Hypertext Preprocessor, more commonly known as just PHP, is a server-side scripting language created specifically for web programming [1]. As a server-side scripting language, PHP code is interpreted by the web server at run-time (when the web page is called for by a client) and not compiled like traditional programming languages such as C++. The server translates the PHP code into Hypertext Markup Language (HTML) each time a page is requested and sends the resulting HTML document to the client computer for the web browser to display. The actual PHP code is not seen by the client, only the HTML [1].

In the nine years since PHP was conceived [1], it has moved from a small scale scripting language used by non-professional web programmers to a powerful language used in large scale enterprise applications [2],[3]. Many professional web programmers are drawn to PHP instead of other similar languages (Perl, ASP and Python to name a

few) because of its specific strengths [1],[3]. First of all PHP is free. It costs nothing to put on a server and use. It is also an open source programming language, meaning that anyone has access to its inner workings and is allowed to modify and add to the language. On top of all that, PHP has a large number of built-in function libraries, giving programmers the ability to do many different tasks without having to create the code themselves [1]. An example of this would be the built-in interfaces with most of the well-known databases (electronic collections of easily searched data), such as Oracle and MySQL [1]. This makes PHP a powerful tool when designing large-scale (and even small-scale) web sites.

Although PHP started as a simple language, it has grown into a language capable of supporting many of the more advanced programming concepts seen today [3]. With the advent of PHP5, the most current release of PHP, the language is capable of supporting almost all of the object-orientated features languages such as C++ and Java are capable of [3]. Unlike previous releases of PHP, PHP5 has the ability to encapsulate data and class member functions, as well as create abstract classes and interfaces [2],[3]. The ability to do object-orientated programming with PHP, as well as procedural programming, opens up a host of possibilities for web programmers seeking to find solutions to problems faced while designing and programming a large scale web site.

3. Common Web Programming Problems:

When designing and programming large-scale web sites there are a host of problems a programmer faces, from the storage, retrieval, and processing of data to the creation of secure environments in which a user can do business [4]. Each of these

problems brings with it unique challenges that the programmer must meet and conquer in an efficient manner if the end goal is a well designed site.

The first of these problems has to do with the storage, retrieval, and processing of data from a database. Most large-scale web sites today have a database somewhere in the background [5]. From insurance companies that give you quotes on auto insurance to large online stores like amazon.com, databases and the information they contain play a foundational role in the functions that these web sites perform. In PHP the ability to connect to these databases, and store or retrieve information is built into the language [1]. The programmer need not worry about writing code that connects to the database and performs the required operations. Instead, the problem the programmer faces when working with databases is the efficient use of the built-in functions. It is all too easy to misuse these functions and create a web site maintenance nightmare [5]. For example, if a programmer had designed a fairly large site (more then ten pages) and had connected each of the pages separately to the database using specific login information, and then at a later date decided to change the database login information, he would have to change the information in each of the pages individually. That would amount to a great deal of work after awhile and would be unreasonable for large-scale web sites.

Another problem web programmers face is that of multiple presentations [5]. It is the idea that there is some data in existence that needs to be presented in a variety of ways. For many large-scale web sites, especially those that seek to meet the needs of users from different countries and languages, the problem exists of presenting data to the user in a way the user understands and can utilize [5]. Often, the web sites of international corporations give users the option of viewing their site in several different

languages. The data, in most cases, stays the same, and it is just how it is presented that changes. The problem for the programmer in these situations is how to present data in a variety of ways and still allow for easy maintenance of the web site. If the information being presented by a web site in multiple languages is indeed the same, and each of the languages is programmed as a separate site, when the data changes each of those sites would need to change. This would make changing data a large project and not one a programmer would undertake lightly. Since one of the greatest benefits of the Internet is the ability to communicate up-to-date information to users, the ability to easily change data is very important.

The last problem that will be looked at here is that of form processing. One of the most common features of web sites are forms [4]. Forms provide a way for users to interact with the web site [4]. They allow information to be taken from the user and used by the web site for everything from searching the site for relevant information, to gathering information about the user, to purchasing items online. There are a variety of ways to handle forms when designing a site. The task of the web programmer is to choose a way of handling them that best serves the rest of the site and minimizes the amount of problems encountered [4]. One of the greatest concerns when dealing with forms is the validation of data passed by the user [4]. Often this data can have direct results on the display of information and the insertion of data into a database and could cause a great many problems if it was not correct. An example of this would be a user signing up to become a member of a certain web site. If they were asked to give their birth date it is quite possible they could enter non-existent numbers, like thirteen for the month. This information would be of no use if it was entered into a database and would

affect the accuracy of the database. Efficient validation of data is a problem that the web programmer must deal with on a regular basis [4].

4. Design Considerations:

When a programmer seeks a solution to a problem, be it interaction with a database or form processing, they have to keep in mind what properties they wish their solution to contain [6]. There are certain principles of good design, such as maintainability, extendibility, readability, and many more, that must be considered [5]. These desired properties, or design principles, affect the value of potential solutions to the programmer. In other words, not every solution is good enough. The final solution to a problem must take all of these principles into account.

One of the foremost considerations when finding a solution to a problem is that of web site maintenance or extendibility [3]. If the problem being faced includes data that needs to change often or requires that new data be added on a regular basis then the best solution the programmer can find for the problem will be one that makes it easy to change and/or add data without affecting the rest of the site [6]. This is also a consideration when dealing with technologies that might change over time. If the web site uses a certain kind of technology to achieve its purpose and a different technology comes out that does the same job better, then it would be a good thing if it were easy to change to the newer method without greatly affecting the rest of the web site's structure [6]. Of course, if the problem is not one that will require the ability to easily change, it is possible to sacrifice maintainability for other design concerns that might not have been achievable if maintenance was the highest priority [6]. The job of the programmer, then,

is to assess the needs of the problem and make conclusions about what properties are going to be most important to have in the final solution. It is their job to decide if they should make maintainability the prime concern or not.

Another concern of web programmers is that of code reuse [3]. It is often the case that code used to solve one problem will work as the solution to another [6]. If when designing a web site, a programmer is faced with several problems of a similar nature, it is possible for the programmer to structure the solution in a way that allows it to be used for all or most of the problems faced [3]. Even if the programmer is only faced with one problem at the time of site design, it is often good to keep the idea of code reuse in mind because of the possibility of future projects in which old code can be used [6]. Keeping code reuse in mind when designing a site requires a little more thinking and possibly more complex designs but in the end it has the possibility of saving time and effort.

Greatly connected to maintainability and code reuse is the idea of modules. A module is a collection of data and operations on that data [6]. Object-Orientated programming is based on the idea of modules [6]. If a solution is designed in the form of a module, with a clearly defined way of interacting with it, it is possible to change the details of the solution without changing the interface, especially if the interface is designed in a very non-detail specific way [3],[7]. Therefore, if many other portions of a site rely of the services of a module, they need not be changed. Modules are also useful for code reuse because they tend to be self-sufficient and easily transferable [6]. Yet, like code reuse, modules add a great deal more complexity to the design and are only worth considering by the programmer if the size of the site is considerable [2]. Often, for smaller sites, the use of modules or object-orientated design is overkill and should be

avoided. For larger sites, on the other had, the added effort of designing modules is worth it [2].

The last design consideration that will be looked at here is the issue of readability. While this is also an issue of good coding practices, it can have a great impact on the design and programming of a site [3]. Large sites are often not the work of one programmer, nor is it guaranteed that the same programmer who designed the site will maintain it [3]. Most of the time it is a team of programmers working together to achieve the goal of a well designed and fully working web site. If a programmer has come up with one of the most ingenious solutions to the problem he was working on but no other programmer can understand what he has done, then, if something in it needs to be changed, it would be nearly impossible to figure out how [3]. A new solution would have to be written. Although this is an extreme example, not being able to understand code another programmer has written without a great deal of work can cause many problems in site maintenance [6]. It is to the benefit of all involved if a programmer designs and programs his code so that it can be understood by others. This, like most other design considerations, takes effort, but also like other design considerations, it is almost always worth it.

The design considerations mentioned here are only a few of the considerations a programmer must take into account when designing and programming a solution to a problem in web site creation [5]. The task that faces the programmer when he must choose which properties he wishes his solution to have is not a small one, especially considering that not all design considerations compliment each other [8]. It is quite possible that in trying to achieve one goal the programmer will defeat another [6].

Combined with all this, the programmer must find a solution that actually fulfills his requirements. This is not a small task.

5. Design Patterns:

The act of designing and programming good solutions to problems faced in web site creation is one that requires a great deal of effort and thought. Finding solutions that meet all the requirements of the programmer can be nearly impossible at times, especially if the programmer is relatively inexperienced when it comes to web programming and the issues surrounding it [8]. Yet most of the problems faced by web programmers are not new. Other programmers have faced them and found solutions [9]. Many times programmers that have had years of experience in web programming have found solutions that could be considered some of the best solutions currently in existence [8]. This is where design patterns come in.

The concept of design patterns is largely credited to the architect Christopher Alexander who wrote several books on patterns in urban planning and building architecture [10]. Although his books were about architecture, the ideas he presented in his books can and have been applied to many other disciplines, including Computer Science. The use of design patterns for software engineering was made popular by the book *Design Patterns: Elements of Reusable Object-Oriented Software* by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides [8],[10].

Put succinctly, design patterns are “named nuggets of insight that convey the essence of a proven solution to a recurring problem within a certain context amidst competing concerns [10].” First of all, patterns describe the essence of a solution, not the

specific implementation. Patterns tell the programmer what they need to know to understand the solution to their problem, yet leave the exact implementation up to the programmer [2]. It is because of this that the same design pattern can be used to solve several problems without ever being done the same way twice [8]. Secondly, design patterns are proven solutions to problems. This is probably the main reason for using design patterns. Most true design patterns have been created by industry professionals with years of programming experience and have been used as solutions to problems countless times before [9]. If used correctly they are proven to work. If this is not true then the pattern in question is not a true design pattern [10]. Lastly, design patterns speak to specific concerns in a specific context. A programmer looking for a solution that fulfills specific design considerations will most likely find a pattern that speaks to most of his needs [8].

The content of a design pattern consists of several elements. While there are some definitions of a pattern that have a great many elements, the most basic are a name, problem description, context, solution, and consequences [9]. The name of a design pattern is a simple and generally agreed upon way of referring to a specific pattern. Having a simple, agreed upon name helps programmers communicate with each other and find information about the pattern they seek to use [8]. The problem describes the exact problem the pattern addresses. With this element a programmer should be able to find a pattern that can be applied to the problem he is facing with relative ease. The context of the pattern describes the conditions that exist or should exist for the pattern to be applied to the specific problem [10]. The solution describes, often with diagrams, the details of the solution to the problem (minus specific implementation information).

Finally, the consequences describe what will happen if the pattern is applied [10]. It shows what the context will be after the pattern has been applied or the benefits and drawbacks of applying the pattern. Many times a pattern description will also include an example of the pattern using a specific programming language [8].

The solutions to problems that design patterns describe are not obvious solutions. They are designs that have required a great deal of thought and effort to create [10]. But they also don't require the programmer to have any knowledge of obscure programming features [9]. Design patterns use common programming practices in unique ways to create elegant solutions to problems. They are a tool programmers can learn to use without much difficulty as long as they know some of the basic programming practices in existence today, such as object-orientated programming.

While design patterns exist for procedural programming, the majority of patterns are linked directly to object-orientated design [8]. They address design issues that come up when using object-orientated programming. Previous to the release of PHP5, programmers using PHP had to take into account the various differences between full object-orientated design and what PHP was capable of [5]. They had to adjust design patterns to work with what they were able to accomplish. Now, with the release of PHP5, it has become much easier to apply patterns to solve problems using PHP [3]. When appropriate, object-orientated techniques combined with design patterns can be used to solve problems frequently encountered by web programmers.

6. An Example:

Suppose that a programmer wanted to create a site that displayed a number in different number bases, such as base 2 (binary), base 16 (hex), as well as base 10 (decimal). The actual data (in this case the number) remains the same no matter what base it is viewed in, but the presentation of the number changes. While this example could be solved easily with a short program, a design pattern will be applied to it in order to get an idea of how design patterns work to solve problems.

The problem the programmer faces in this case is that of multiple presentations [5]. The data and the operations on the data stay the same while the presentation of the data changes. The goal, then, is to identify a design pattern that solves this particular problem. In this case the applicable design pattern is Model-View-Controller (MVC) [11].

The MVC pattern is concerned with the presentation of data that is always changing or requires multiple views. The solution it gives for this problem is to split the site or application into three separate parts; the model, the view, and the controller [12].

The model consists of the data and all the operations that are performed on this data. Everything that has to do with data manipulation is kept in the model [11]. Ideally, the model is completely separate from the actually presentation [12]. It contains no information of how the data is to be presented to the user on the web site. In our example, the model would store the number we are concerned with and all the functions that convert the number to different bases.

The model also has a well-defined interface with the outside world that allows it to give its data to others and perform requested operations on the data [11]. In our

example the model contains a function called returnNumber() that passes the number in a specific base to whomever requested it.

The code for the model would look something like this:

```
<?PHP
	class theModel {

		private $theNumber = 16;

		function returnNumber($base)
		{
			switch ( $base )
			{
			case 2:
				return $this->baseTwo();
				break;
			case 10:
				return $this->baseTen();
				break;
			case 16:
				return $this->baseSixteen();
				break;
			default:
				return $this->baseTen();
				break;
			}
		}

		private function baseTwo()
		{
			// convert number to base Two and return
		}

		private function baseTen()
		{
			// convert number to base Ten and return
		}

		private function baseSixteen()
		{
			// convert number to base Sixteen and return
		}
	} // end of theModel class
?>
```

The next component of the MVC pattern is the view. The view is the part of the solution that is concerned with the presentation of data [12]. It has nothing to do with storage and manipulation of data, only with presentation. The view is what the user sees and interacts with. In our example, the view takes a number and tells which base it is in before displaying the number as well. Like the model, the view has a well-defined interface. In our example, the view has one function, displayNumber(), that can be called to display a number in a certain base to the user.

The code for the view would look like this:

```
<?PHP
        class theView
        {
                function displayNumber($number, $base)
                {
                        switch ($base)
                        {
                        case 2:
                                echo “This is a number in base 2:”.$number;
                                break;
                        case 10:
                                echo “This is a number in base 10:”.$number;
                                break;
                        case 16:
                                echo “This is a number in base 16:”.$number;
                                break;
                        default:
                                return $this->baseTen();
                        break;
                        }

                }

        }
?>
```

The last component of the pattern is the controller. The controller exists to manage requests from the user [12]. It receives information about what the user wishes to

do and calls the appropriate functions in the model and view that will accomplish the user's request [5]. In our example, the controller calls the appropriate functions in the model and view to display the number in the base requested by the user.

The code for the controller would look like this:

```
<?PHP
        // In another page there will be HTML code that creates a form asking
        // which base the user would like to see the number in. The form calls
        // the page with the controller to decide what action to take.

        $model = new theModel;
        $view = newView;

        switch ( $user_choice )
        {
        case 2:
                $number = $model->returnNumber(2);
                $view->displayNumber($number, 2);
                break;
        case 10:
                $number = $model->returnNumber(10);
                $view->displayNumber($number, 10);
                break;
        case 16:
                $number = $model->returnNumber(16);
                $view->displayNumber($number, 16);
                break;
        default:
                $number = $model->returnNumber(10);
                $view->displayNumber($number, 10);
                break;
        }
?>
```

Using this pattern the programmer can achieve a high degree of separation between the storage and manipulation of data and the presentation of data [12]. This makes certain kinds of maintenance easy. If the programmer decided to change they way he converted a number into a different base, he would only have to change the model. The view would never be touched. Similarly, if the view needed to be changed, the model

could be left alone. This makes the MVC pattern a very powerful tool, but it also makes using it more difficult and adds complexity to the site design [12]. It isn't a pattern for small-scale web sites. Large-scale web sites, on the other hand, could gain a great deal from MVC [12].

7. Conclusions:

Designing web sites presents a number of problems that a programmer needs to find solutions to while also keeping in mind good design principles. This is a large and complex task. Yet many of the problems web programmers face are common and have been dealt with before by industry professionals. It is possible to gain from the experience of others by way of design patterns. Design patterns present the heart of a well-proven solution to a specific problem in a non-implementation specific way that allows for a programmer to use the solution to solve their own web design problems. They don't have to find the solution for themselves. Programmers can rely on others and get the job done efficiently and well. This is why design patterns are a valuable tool for web programmers if they are willing to take the time to learn how to use them.

8. References:

[1] L. Welling, and L. Thomson, *PHP and MySQL Web Development*. Indianapolis: Sams Publishing, 2003.

[2] J. Lam, "Introduction to Design Patterns Using PHP," 2003, http://www.devarticles.com/c/a/PHP/Introduction-to-Design-Patterns-Using-PHP/

[3] G. Schlossnagle, *Advanced PHP Programming*. Indianapolis: Sams Publishing, 2004.

[4] N. Wallace, "Design Patterns in Web Programming," 2000, http://www.e-gineer.com/articles/design-patterns-in-web-programming.phtml.

[5] P. A. J. Braam, "Design Patterns applied to Web Programming in PHP," 2004, http://www.cs.vu.nl/~pajbraam/Essay_OOP.pdf.

[6] S. McConnell, *Code Complete.* Redmond, Washington: Microsoft Press, 1993.

[7] H. Fuecks, "Guidelines for Designing Classes in PHP," 2003, http://www.phppatterns.com/index.php/article/articleview/31/1/1.

[8] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1995.

[9] L. Atkinson, "Applying Patterns to PHP," 2001, http://www.zend.com/zend/trick/tricks-app-patt-php.php.

[10] B. Appleton, "Patterns and Software: Essential Concepts and Terminology," 2000, http://www.cmcrossroads.com/bradapp/docs/pattern-intro.html.

[11] H. Fuecks, "Model View Controller Pattern," 2002, http://www.phppatterns.com/index.php/article/articleview/11/.

[12] B. Kotek, "MVC design pattern brings about better organization and code reuse," 2002, http://builder.com.com/5102-6386-1049862.html.