FileMaker API for PHP

A Practical Guide for Creating Database Driven Web Sites with FileMaker Pro 11 and FileMaker Server 11

Todd Duell

Table of Contents

Part VII — Publishing FileMaker Data with PHP

Part VIII — Additional Information

Introduction

Welcome, and thank you for acquiring FileMaker API for PHP, A Practical Guide for Creating Database Driven Web Sites with FileMaker Pro 11 and FileMaker Server 11. If you're reading this book I'm acutely aware that you have been challenged to publish data from your FileMaker Pro database to the web. Many of you have no web publishing experience, have experience on other platforms and technologies, and/or are frustrated by the lack of documentation, examples, third party products, and available rapid application development tools to publishing data from a FileMaker database. The FileMaker API for PHP has only been available since 2007. Therefore, I will assume that you are at least new to the FileMaker API for PHP and are seeking out resources to assist you to publish your data to the Internet in an efficient and professional manner.

This books starts by demonstrating basic HTML that you should already be familiar with. If you are not already familiar with HTML you should seek out a book or other on-line resource to help you understand how to work with basic tables, forms, links, etc. When publishing data from a database you will be required to know how to create the tags such as headers, table rows, and table data to display your data correctly. If you can master this, you can create a dynamic web page to display your data. I don't want to scare you right up front. It's actually quite easy. However, like anything else, it does take practice and vision to know how you want the page to display.

Section II of the book deals with enabling either Apache on OS X or IIS Server on Windows. Both platforms support PHP natively. The nice part is that the FileMaker Server installation process will install PHP and turn everything on (for the most part). Your choice is simply to decide which platform you want to use to host your web site. The most important consideration for a server it to develop on the same platform on which your web site will be hosted, weather that be an in-house server or a third party hosted server. Although it's not required, it does make testing and deployment that much easier.

Part III discusses your FileMaker Server installation. If you were wondering, you only need FileMaker Server to host your solution, not FileMaker Server Advanced. That saves you quite a bit of money. Unfortunately, you can't use FileMaker Pro or FileMaker Advanced as a development tool when accessing the data via PHP. So you will have to break down and purchase FileMaker Server.

In Part IV, this book covers the use of rapid application development (RAD) tools versus "free" text editors. You have to develop your web pages in some type of text editor. The question is how much money are you willing to spend to improve your productivity and extensibility. There are definite advantages to utilizing professional tools.

Part V is where we start to dig in. It's imperative that you learn some basic PHP. I'm not going to pretend to be an expert PHP developer as there is just too much to learn on top of everything else. However, a little bit of knowledge goes a long way. As in any programming language, you use maybe 80% of the available methods on a routine basis and you look up the other 20% only when you need to use them. This book is going to make you hand code the PHP. Yes, I can hear you grumbling. The problem is that there are no complete RAD tools to do everything that you need for professional solutions. So it's best learn it the hard way the first time, then use the tools that are available to supplement your work.

Part VI is an overview of the provided sample database. It does not go into excruciating detail about how to create a FileMaker Pro database. I'm going to assume that you already know how to do that. The database is very simple. It is used to demonstrate specific techniques for acquiring data from layouts, fields, value lists, portals, and how to run scripts. It does cover every possible scenario that you would build in a commercial application. I believe that it's better to focus on the PHP methods rather than throwing a curve ball at you with unnecessary, complicated database structure or an esoteric topic. If you have a simple database to start with you can apply the same PHP code to far more complicated systems.

If you are already comfortable with building a FileMaker Pro database and PHP you can skip right to Part VII. The sample files start off slow specifically for beginners. They demonstrate all the basic techniques to search, display, add, edit, and delete records. Then they graduate into more complex PHP and FileMaker methods to do things such as display container images, sessions, login routines, email, running scripts, integrating with Google web services, etc. The main focus of this book is to provide complete, professional, production quality examples. You should be able to take these techniques and use them immediately in your own work. I will point out that there are many ways to accomplish the same end result. As such the sample files will create the content using a variety of methods so you can pick and chose the one that fits your programming style the best. The more techniques you learn, the better the programmer you will become.

Finally the book wraps up section VIII with reference material to other web publishing technologies that you can use with FileMaker Server and Server Advanced. Next is the cost, which is always an important consideration. Then topics to consider for performance and security expectations. Lastly, where to find more information about logs and the FileMaker API for PHP reference guide as a last resort when things go wrong.

There are several conventions used in this book:

Localhost

Local host refers to your development server. The URLwill typically be http://localhost/

Production Server

Production Server refers to your web server that is hosting the solution to the Internet. The URL will typically be http://www.mycompany.com

Code Samples

PHP code will be displayed in `Courier` font to make separating the code from the discussion easier.

```
<?php echo 'This is my PHP code'; ?>
```

Special Content

Special content or discussion will be called out to your attention.

Notes specific to the use of the FileMaker database will display the FileMaker Advanced icon.

Notes specific to the use of FileMaker Server will display the FileMaker Server icon.

Notes specific to PHP methods will display the PHP icon.

Notes not specific to anything will display the notes icon. Notes present an interesting topic that you can research further on your own.

Specific precautions that you should pay attention to. If you don't you are bound to run into problems.

About the Author

Todd Duell is the CIO of Formulations Pro, Inc. and has been creating powerful custom and commercial solutions using FileMaker Pro since 1989. Todd holds an M.B.A. in Technology Management, is a Certified FileMaker Pro Developer, and has been a member of the FileMaker Business Alliance since 1998. Todd has published more than 300 technical articles, white papers, and open sources databases to the scientific and Formulations Pro community of more than 3,000 worldwide subscribers. In Todd's tenure at Formulations Pro since 1997, he has created more than 38 commercial database and web-based applications for numerous worldwide, multi-billion dollar companies. Additionally, Todd is a regular contributor to Advisor Magazine, the industry-leading trade magazine for the FileMaker Developer community, and for the FileMaker Technet Resource Library at FileMaker Inc. Lastly, Todd frequently speaks at conferences and seminars on topics ranging from 21 CFR 11 (electronic records and signatures) to HIPAA compliance using FileMaker technology.

On a personal side, Todd has been an avid Triathlete for the past 20 years, competing in more than 200 races with more than 100 top-3 age group finishes and numerous top-10 overall finishes. Todd has also finished 4 Ironman events, including the famed course at Kona, Hawaii. When he's not training or racing, Todd can be found playing outfield for Magic Sports — a "Majors Plus" rated men's tournament softball team. This is the highest worldwide ranking a team can achieve. The teams that Todd has played for have played at the State Championships and World Series of Softball for a variety of associations every year since 2000. In 2010, Magic Sports won both the SSUSA California State Championship and the World Series. Todd was named to the "All Tournament Team", batting an impressive 0.760 average.

Part I — Getting Ready for Web Publishing

Creating web pages requires that you have at least a basic working knowledge of HTML. If you are already familiar with HTML you can skip Part I. Otherwise, I'll cover just the basic HTML elements that will be included in all the sample files. That way you won't encounter any surprises and will be able to refer back to this section if you get stuck for any reason.

One of the things you'll notice right away in the sample files is the simplicity of the code. There are no graphics or design, very little page structure, and the only apparent use of HTML is to display the data from the database. This is for a reason. I don't want you to get stuck trying to weed through a complex page with images, CSS, and other elements. I want you to be able to focus solely on the PHP code to display information from the database. Once you learn the PHP code, the layout and design is easy. For now, the only special thing you need to know is how to display a PHP document. The file needs to have the extension .php. That tells your web server to process the PHP code on the page.

Create a Table

Data has to be placed into a table so it can be displayed and organized in a logical fashion. Without tables all your data will run together and end up as a big mess. Take a look at the sample code below and how it will display in a browser in Figure 1. You'll see that the code creates a basic table with 2 columns and 2 rows. As in all HTML and PHP documents you have to start the document with the tags: <html>, <head>, <title>, and <body>. The <html> tag tells the web server to process anything in the document and present it as html to the browser. The <head> tag contains the title of the document that displays at the top of the web page. In this case I named the page "Sample Table". Then I closed the head section by using the tag </head>. All tags are closed by using the forward slash "/" with the tag name. Don't forget to close the tags in the same order in which they are created or you will get some strange results.

```
<html>
<head>
<title>Sample Table</title>
</head>
<body>
 <table border="1">
      <tr>
            <th>First Name</th>
            <th>Last Name</th>
      </tr>
      <tr>
            <td>Todd</td>
            <td>Duell</td>
      </tr>
 </table>
</body>
</html>
```

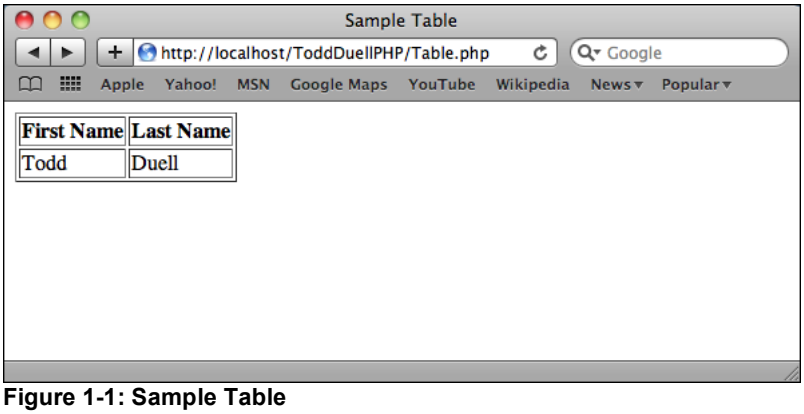


Figure 1-1: Sample Table

Next is the <body> tag. This is where all the content for the web page is created and displayed. It will be your job to create the PHP code to output the correct html structure. For a simple table, which is used 99% of the time, you'll start with the <table> tag. Notice I included the attribute border="1". I did this because I wanted to table to draw a border with a line thickness of 1 pixel. There are a number of standard attributes that you can utilize to format the cells, including cell padding and cell spacing.

Most of the time you'll want to create a header row using the field names. To do this I created the first row with the <tr> tag. Then displayed the First Name and Last Name headers using the table header <th> tags. The <th> tag will display your headers with emphasis (i.e. bold) without actually having to use the tag. Don't forget to close the first row with the </tr> tag. Then move on to the content using the same process as above. Except this time, the content will use the table data <td> tag to display the table data (i.e. record data from the database) as plain text. Once you're done with the table row you can close the tag as well as the tags for the table, body, and html tags.

The lesson here isn't how to create a table, table rows, table header, table data, etc. The lesson is to observe the repetitive nature of displaying HTML data. To put it in perspective; each record in the database will be wrapped in a <tr> tag. Each field will be wrapped in a <td> tag. While you are looping through records and fields your PHP code will simply add the start and stop tags to the data from the database. That will enable a browser to correctly display the content.

Create a Form

Forms will quickly become your best friend when working with data from a database. You will use forms to search for records and to submit data to create, update, and delete records. Inside the form you will be able to pass various pieces of information to the web server and database. The data can either be user entered, as in the Search field shown in Figure 2, or it can be hidden information that you don't want the user to be able to see.

All forms use the <form> tag. The action attribute is the path to the document that you want to process the form. It can either be an absolute path such as http://www.mysite.com/Form.php or it can be a relative path to another document in another folder such as ../DifferentForm.php. Sometimes the form will point to itself and sometimes it will point to a different file. It really depends on how you want to set up your code for the user workflow. There are distinct advantages to both methods. Pointing the action to the same document means that you'll only have one document to manage. However, the code on that one document may be quite long to handle a variety of branching user interactions. Pointing the action to a second document will allow you to process the form in a more discrete, logical fashion — similar to a sub-script in FileMaker Pro. The downfall is that you'll have two pages in your site to manage.

```
<form action="Form.php" method="get">
<input name="id" type="hidden" value="<?php echo $_GET['id']; ?>">
Search: <input name="Search" type="text">
<p>
<input name="Search" type="submit" value="Submit">
</form>
```

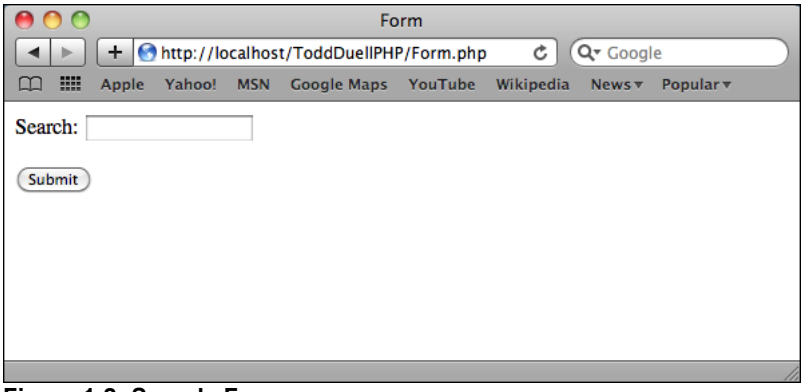


Figure 1-2: Sample Form

The method attribute is of critical importance. There are two methods that you can use to pass information between web pages — GET and POST. GET will send all the data from the form in the URL. The URL is visible to the users. The advantage to sending the data in the URL is that the users will be able to bookmark your page.

The disadvantage is that the users may be able to see information you don't want them to be able to see. Additionally, there is a limit to how much data you can send in the URL. You can only send 2048 characters. That may seem like a lot, but when you are creating and modifying records, 2048 characters can get used up very quickly. Using the GET method is useful mostly for searches.

The other method is POST. POST information is sent in the header instead of the URL. That means that you can hide information from the user and send much larger amounts of data. Just because the user can't see the data doesn't mean it's secure. For real security, you still need to use an encrypted page (i.e. https). POST is useful for creating, updating, and deleting records.

Both GET and POST data will be sent using Name=Value pairs. That means that you'll define the name of the value, such as the field name. Then supply something like a text field for the user to enter the value. For example, in Figure 2 if I typed in Todd in the Search box the Name=Value pair would be Search=Todd. To extend the topic of Name=Value pairs, the actual URL from the example form will look like this:

```
http://localhost/ToddDuellPHP/Form.php?id=123&Search=Todd&Search=Submit
```

It starts with the domain and web folder of the specified file from the form action attribute: http://localhost/ToddDuellPHP/Form.php. Then it adds on all the Name=Value pairs defined in the form. In the example form there are three Name=Value pairs. The first one is a hidden value for the Record ID (id) and the value obtained from the existing URL name of 'id': id=123. Then the form concatenates the next values using the & symbol. The next Name=Value pair is Search=Todd, or whatever you typed into the Search text box. Finally, the button that is pressed by the user will also be sent. In this case the Name=Value pair for the button is Search=Submit. In the example files you will see how important it is to name your buttons to determine exactly which button the user pressed. Especially when you have more than one button on your form.

Now let's back up for a moment and take a close look at what you can put inside your form. In this example I've placed the hidden element of "id" in the form specified by type="hidden". Hidden values allow you to pass information to the web server and database without the user actually seeing any data on the browser screen. It's very useful for data such as Record ID's. In the example I'm passing some PHP code that retrieves the URL value from 'id' with the $_GET superglobal method.

```
<input name="id" type="hidden" value="<?php echo $_GET['id']; ?>">
```

The other item I'm passing in the URL is the Search content from the Search text field. I've given the search field the name 'Search' and defined it as a text field that the user can enter. There are also other user enterable fields for text area, check box, radio button, list/menu, etc. that will be demonstrated in the example files later in the book. For now, it's only important to know that the data in the form will all be sent to the web server as a Name=Value pair.

```
Search: <input name="Search" type="text">
```

Finally, you have to create a button to submit the form to the web server. The name in the example is Search and the value is defined as Submit. Those values are also passed to the web server. It's very similar to a FileMaker Pro script dialog box where you have 3 buttons to choose from. When the user clicks a button you check for the button they pressed by using the function Get(LastMessageChoice). In the case of a form, you'll be checking for Search==Submit, which is probably a lot more intuitive than checking for Get(LastMessageChoice)=1, 2, or 3. I always hate having to look at the Show Message dialog box to figure out which button is for which response, especially when the default button is something other than "OK".

```
<input name="Search" type="submit" value="Submit">
```

Create a Link

Links allow users to navigate from one page to another page, within the same page, or even pass information to the web server to perform processes such as search for records. You probably recognize a link from seeing "Click Here" on a web page. Rather than displaying the absolute or relative link to that location the web page usually shows a text or image-based link. Links with databases are useful for a variety of items. In the example files we'll be creating links to navigate to detail pages based on the Record ID and links to change the sort order of the records by changing the URL.

All links start with the anchor tag , where the value between the quotes for href is a relative or absolute path to a document. href stands for hypertext reference. Basically, it’s saying what’s the path to the document you want to link to? Next, put whatever text you want to display on the web browser (i.e. Click Here). Then close the anchor tag . That’s it. Links are really easy.

```
<a href="3_0_SearchRecordsSimple.php?FirstName=Todd">Click Here</a>
```

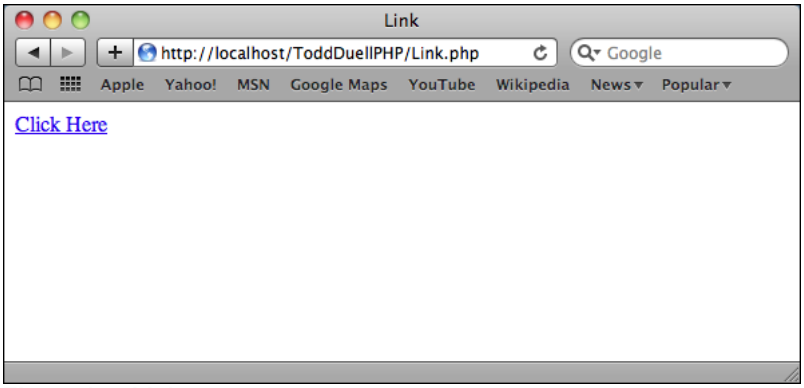


Figure 1-3: Link

Meta Refresh and Redirect

Meta refresh is also known as a redirect. It’s used to refresh or reload the current page or redirect the user to a different page after a defined amount of time. This is usually installed on a web site if pages have moved or your PHP code has finished and you want to have the user redirected to the starting page in the workflow.

To refresh your page every 3 seconds you can use the following code. You probably don’t want to refresh your page very frequently because it does make a new call to the web server each time. I think you can image the load it would put on the server if hundreds or thousands of users were refreshing their pages every few seconds. That’s where AJAX (Asynchronous Java Script) is probably more useful because it can refresh just certain sections of your web page.

```
<meta http-equiv=”refresh” content=”3” />
```

If you want the redirect to happen at a certain time interval and send the user to a specific URL you can use the following code. The URL, of course, can be either a relative or absolute path.

```
<meta http-equiv=”refresh” content=”3;URL=4_0_SearchResults.php” />
```

Or if you want to immediately redirect the user you only need to provide the following code without any other tags on the page (i.e. html, head, body, etc.). Just substitute the page (location) you want to go to.

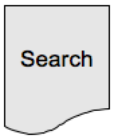
```
<? PHP
header("Location:1_0_FindAllRecords.php");
?>
```

Designing a Web Application

Designing a web application is unlike anything you’ve probably had to do when compared to developing a FileMaker Pro database. FileMaker makes most of the processes so seamless that you really don’t want to know what goes on under the hood. As a FileMaker developer you’ve become accustomed to making one layout that does absolutely everything — search, create new records, modify records, delete records, etc. Well let’s just say that a web application isn’t quite as smart as FileMaker. As a stateless process, meaning there is no constant connection to the database server, you have to tell (program) the web pages absolutely everything they need to know. That means that you have to create all the necessary web pages for every single workflow. In many cases the workflows really can’t be re-used. For example, to accommodate one FileMaker layout and all the things that it can do, you’ll have to make no less than four web pages! If you have portals on your layout, you’ll need to double that number. The good news is that there are standard methods and nomenclature to describe the workflow.

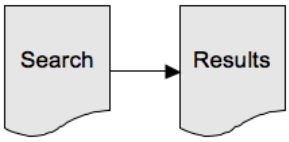
Once you grasp the workflow and the nomenclature you can communicate and create any process for your users very quickly.

Search: The Search page is designed to search and display all the records on one web page.

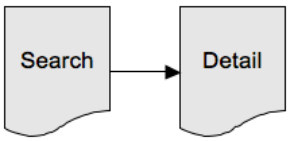


List: The List page is designed to display a pre-defined record set on one web page.

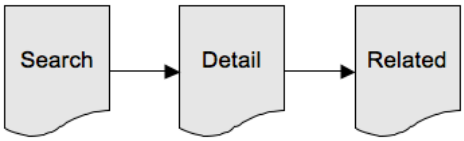
Search>Results: This method is designed to search for records on the Search page. Then display the records on the Results page.



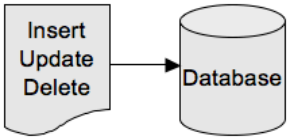
Search>Detail: This method is designed to search for records on the Search page and display a limited amount of information. The user can then click a link to display additional content on the Detail page.



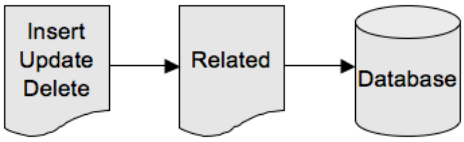
Search>Detail>Related: This method is designed to search for records on the Search page and display a limited amount of information. The user can then click a link to display additional content on the Detail page. The Detail page may also contain related records, which the user can click to display additional related content on the Related page.



Insert, Update, Delete>Database: There are three ways to modify a record. You'll need three separate pages to perform each task. Insert is the same a creating a new record. Update is the same as editing a record. Delete obviously deletes a record. Unless you create very complex PHP code or run a script on FileMaker Server to process more than one record at a time, the FileMaker API for PHP can only execute on one record at a time.



Insert, Update, Delete>Related>Database: If you have to work with related records you'll need to start on the parent record. Then click the related record to isolate it as a single related record before you can process the record in the database.



Index

N

O

P

Q

R

S

FileMaker API for PHP

A Practical Guide for Creating Database Driven Web Sites with FileMaker Pro 11 and FileMaker Server 11

The only complete reference book for the FileMaker API for PHP using FileMaker Pro 11 and FileMaker Server 11. Whether you are a beginner or a professional developer you will find the information in this book to be invaluable. Part I starts by getting you ready to publish data to the web with a primer about the basic HTML you will need to know to work with the examples in this book. Part II shows you how to install and configure either Apache or IIS as your web server. Part III shows you how to install and configure FileMaker Server. Part IV provides information about free and professional text editing tools to build your web sites. Part V teaches you the basics about PHP syntax to work with the examples in this book. Part VI explains the sample database fields, layouts, and scripts that are used for all the examples in the book. Part VII contains more than 40 complete PHP examples that will teach you techniques to build professional web sites. Part VIII wraps up the book reference information with extended FileMaker vs. PHP syntax, other web publishing technologies available to FileMaker, expected costs, performance and security considerations, understanding server logs and error codes, and a FileMaker API for PHP quick reference guide.

Examples Files Include:

- How to connect to the database
- How to display images from a container field
- Find records
- Perform compound searches
- Sort records
- Insert records
- Duplicate records
- Delete records
- Validate the data before inserting or updating records
- Working with related records in portals
- Uploading files to the server
- Creating list menus
- Displaying search results
- Create a login routine to secure your web pages
- Running scripts using PHP code
- Importing records after uploading an Excel file
- Display record pagination and record navigation links
- Working with multi-user web pages
- Display Boolean and multi value check boxes
- Display multi value list
- Quick find method
- Google reCAPTCHA
- Display random banner ads
- Bonus files with sample code

For more information visit us at:
www.formulationspro.com

Contact the author Todd Duell at:
tduell@formulationspro.com

ISBN-13: 978-0-61543-681-1
ISBN-10: 0-61543-681-1