FileMaker Server
Accessing your databases in a web browser

With SUNRISE Contacts 2017 and a copy of FileMaker Server, you can access the data in your databases in a web browser. There are two ways you can do this:

1. WebDirect.
2. PHP/XML/REST

WebDirect

By far the easiest and most powerful is a technology known as WebDirect. It is the fancy new marketing name from Apple Inc. for *web publishing* databases on the web.

In FileMaker Pro 12 and earlier, it was more correctly termed Instant Web Publishing (IWP). Now with WebDirect, you get the same feature, just a little more refined (and uses the latest HTML5 for more accurate reproduction of layouts in a web browser). However, with this new refinement and fancy name change has meant that web publishing is no longer free (or cost you no more than purchasing a copy of FileMaker Pro standard at around $300), even for a limited number of concurrent users accessing your database in a web browser.

There was a time when you could access your databases online without purchasing a FileMaker Pro app. Once you create the Runtime app (from which SUNRISE Contacts 2017 is built), virtually anyone could use IWP in a limited way[3] to let you access the databases and present the layouts in a reasonable way for free. Even if the feature could have been made available only for FileMaker Pro at a cost of $300, it was still not enough to keep Apple happy. The problem is that the company could not make enough money from selling enough FileMaker Pro apps to other users who may want to access the databases in a web browser. Apple, being a shareholder company, must find ways to please its shareholders with substantial profits. Furthermore, the person who owns the databases can potentially bypass all the push technology when accessing his/her own data. Not good for Apple after spending more than a billion dollars building its own servers in the United States and with a goldmine of potentially useful information coming from users and business professionals being siphoned away by the FileMaker platform should users choose to go this way. So, after much consideration, the company has come up with a new and presumably better solution. In return for a refined and better web publishing feature under a new fancy marketing name of WebDirect, you must now pay a monthly subscription (if hosted on the internet with no known number of anonymous users) or per concurrent user (for companies

[3] Up to 5 concurrent users (originally 9 in earlier versions until Apple came along and bought the licensing rights and source code to the software from Claris Corporation) could access FileMaker databases in a web browser. Even with 5 users, this was still a useful feature for small groups within an organization or for a family.

that use WebDirect in-house) fee-based model. In this way, Apple can grab a continuous slice of the economic pie from individuals and organisations that decide to use WebDirect.

The advantage of IWP/WebDirect is the means by which users can view the design of layouts and all the data held in the database in any web browser without ever needing to learn HTML, PHP, Javascript or anything else. Each layout in a database is essentially a web page and is rendered as such by WebDirect in a remarkably accurate way. Data presented in a web page can be changed easily or data can be obtained from users in a form and stored using the database engine. And with a built in server, the web page and all the data can be delivered and published on web browsers for any user to access.

Indeed, this is the future of web publishing for all non-geeky people. The ultimate aim is to create a software tool that allow users to build and design their web pages through a simple drag-n'-drop approach, have a built-in database to present and store data through the web page, and a means by which to serve the web pages to users. At last, the age is upon us to literally do away with learning any new web programming language and deliver everything on the internet in a very simple way. This is possible thanks to WebDirect and FileMaker Server (although you still have to purchase FileMaker Pro to build the layouts). However, there is only one tiny catch: Apple wants to make a continuous and hefty profit from anyone who chooses to use WebDirect to publish online.[4]

How to enable WebDirect

If you are not perturbed by such restrictions and higher costs and would like to pursue web publishing through the WebDirect option, here are the steps to achieving this:

A. How to host SUNRISE Contacts 2017 through FileMaker Server

1. If you have FileMaker Server 16 or higher, you have to prepare your databases for hosting on the server. You can do this with FileMaker Pro 16, or we have saved you the cost by using SUNRISE Contacts 2017.app/exe. What does "prepare" mean? This is nothing more than specifying which databases should be visible in a web browser (we have done the hard work of creating the databases), entering a couple of other details (supplied by the people who give you access to the internet called ICT staff

[4] To reduce consumer interest in IWP, a bug was introduced into IWP in FileMaker Pro 11 making it difficult to enable this service. The aim was clear: at some point Apple wanted to drop the feature altogether if it thinks not enough users would complain about it. However, interest in the feature remained strong. So Apple had to continue supporting it. Now a solution has been found to give people what they want, but ensure the company makes a profit in doing so. It is called WebDirect, and you must pay for the privilege of accessing your own databases on the web using this technology. Unfortunately the payment model is designed in such a way that it would be too expensive for consumers to enjoy the benefits of SUNRISE Contacts 2017 in a web browser. Only big companies with money to burn can afford this option.

members or your ISP) such as IP address and port number, and then send the databases to FileMaker Server 16. The only other preparing work you must do is at the FileMaker Server 16 side. Here, you must add the specific fonts you use to display text in the databases to the server's own font folder. By doing this, printing the layouts of the database on a web browser will ensure the text will appear exactly as they do in FileMaker Pro on a client's machine.

2. As part of the preparation process, you might be wondering, "Which databases should I make visible for hosting on a network?" You can always choose to show all the databases to your online users. But for greater security, it is usually better to hide databases that you know users will never need to access. Remember, hiding does not necessarily mean no access. It only means users cannot see other databases, but the ones that are visible can still access those databases. For example, contacts.fmp12 requires postcode.fmp12 to obtain postcode information. This kind of information would probably be useful to view by online users if you choose contacts.fmp12 to be visible to your users. Therefore, contacts.fmp12 can be made visible online, but postcodes.fmp12 can be hidden from view on all web browsers while still allowing contacts.fmp12 to obtain postcode data. To simplify the process of knowing which databases are linked together for the Contacts, we recommend that you drag-and-drop the entire folder of core (or standard) SUNRISE Contacts 2017 databases into the FileMaker Server database hosting folder (ignore the self-running application supplied with SUNRISE Contacts 2017 known as *SUNRISE Contacts 2017.app/exe*, this can be removed and serves no purpose in FileMaker Server). When you have the databases in the Server's folder, make visible only contacts.fmp12. This should be more than enough to make it work.

3. Add any other lookup databases you may have purchased from the SUNRISE web site to the SUNRISE Contacts 2017 folder contained inside the Server's database hosting folder and make them visible if you wish other users (including yourself) to access them online.

4. Launch FileMaker Server (if you have not done so already).

5. Let FileMaker Server know the location of the SUNRISE Contacts 2017 databases for hosting. This is basically the folder you have set up in FileMaker Server to host databases online. If you have dropped SUNRISE Contacts databases into this hosting folder, they will be hosted by FileMaker Server.

6. Once FileMaker Server knows you have the databases ready for hosting, click the button in FileMaker Server to start up web publishing. It will immediately start serving the databases online. You are ready to open a web browser on another computer to see the results.

B. Configuring for WebDirect using SUNRISE Contacts 2017.app or FileMaker Pro.app

1. Don't have FileMaker Server but still want to begin the process of "preparing" the databases for serving by FileMaker Server? No problems. You can do this directly through SUNRISE Contacts 2017.app or FileMaker Pro.app (version 16 or higher). Just open up contacts.fmp12 (or launch SUNRISE Contacts 2017.app/exe). Next, we recommend that you open the Lookup Databases layout (press Command or Ctrl 8) to at least give the app an opportunity to open up and check all the available databases (including the ones that will be hidden). Now go back to another layout to show the expanded File menu as needed for step 2.

2. Under the File Menu, you will see extra commands. Choose Sharing → Configure for FileMaker WebDirect. From there you will see all available open databases and options on how to configure the databases to allow sharing (i.e., for other users to see and have access to the databases, or be hidden but still allow other databases that are linked to certain databases to have access to them).

3. When configuring your databases, you have the option to turn off sharing and hide the databases from other users. To do this, select the database(s) in the left pane. Use the Command (or Ctrl) key as you click more than one database to highlight them. On the right side, click the radio button that says "No user". Then, to make sure the database is never shown in the Launch Center or web browser for other users, put a tick in the check box that says "Don't display in FileMaker WebDirect Launch Center".

4. However, if you do want to web publish your database(s) using WebDirect to allow users access to them, select the database(s) as you would in step 3. Put a tick against one or more databases you want to web publish (i.e., you want to see, or make visible, the layouts and data in a web browser via WebDirect). Choose "All users", or select "Specify users by privilege set" and place a tick against the privilege set to allow users access to the database(s) under those privilege sets. Once you are finished and press OK to finish off the configuration, you can upload the database(s) individually using File→Sharing→Upload to FileMaker Server. Or you can manually transfer the databases you want to share straight to the FileMaker Server hosting folder, whichever is convenient for you. But before you do, we recommend you read the remaining steps below to decide if you need to do anything else to your database.

5. Step 4 allows users to see the shared database(s) in the Launch Center (via the web browser). However, if one or more databases need access to data in another database but you don't want the

other database to be visible, select the other database and put a tick in the check box that says “Don’t display in FileMaker WebDirect Launch Center”.

6. Close the Sharing dialog box.

7. If you have selected users to access shared databases by privilege set, choose “Access…” in the File menu. In the Database Access layout, you need to make sure the users who will have access to the database are shown in the records. (this is another level of security to ensure only those users are allowed access to the database. If not, those users will never be able to login and access the database through FileMaker Server (even if the database is visible in a web browser). So to give users access, create records for each user or groups of users based on authentication account group (either Admin, FileMaker Guest, or [external guest]). If this is the first time you have seen this layout, the default users who can access the database are FileMaker Guest and Admin (by default, the Owner user has immediate access to the database, so this user does not need to be shown, but you can if you wish have it shown). Thus removing these default records will stop FileMaker Guest and Admin users from gaining access to the shared database. So decide carefully whether to keep or remove these records. If you are using FileMaker Server and want guest users to use their own external authentication (i.e., the one they use to log onto their computers), create new records for all these users in the layout. Thus, in the Account Name field, enter the username of each user (the one he/she will enter to log onto his/her computer) like so *Guest//Username*. Additional fields shown in the Access layout called Full Name, Position, Work Phone, Room and Email are there to help the administrator/owner (i.e., yourself) to contact users quickly and identify who is accessing the database. Finally, the Status field tells the administrator/owner of any problems for specific users, such as login failures. If you want the same users to access other databases, the records of this Access layout should be exported (i.e., you should see a file called *access.exp*). Rename this file to *access.imp*, and import the records into the same Database Access layout of other databases to help speed up data entry work.

8. The databases of SUNRISE Contacts 2017 are designed to auto-login to the Owner account. For the protection of your data, choose File→Change Password to change the password. Use File→Manage→Re-Login to login to another account (i.e., Admin or FileMaker Guest) and change the password for those accounts (if you have direct access to the databases). You can also change the password through File→Accounts and select the account name and press the Reset Password. Then you must re-login to the account to change the password.

9. Close the databases. The information you have specified in the Sharing dialog box and Database Access layout will be saved with

the databases.

10. Now you are ready to publish your databases to the web. You can either drag-and-drop the entire folder of databases (once you have properly quit the application for preparing your databases) into the FileMaker Server database hosting folder (without SUNRISE Contacts 2017.app/exe), or you can individually upload each database file to FileMaker Server using File→Sharing→Upload to FileMaker Server…

C. Web browser access to SUNRISE Contacts 2017

1. If your database(s) are being hosted by FileMaker Server, it is time to open your web browser.

2. Type in the address: http://127.0.0.1/ (or whatever web address is specified by your administrator). If you have changed the port number (the administrator will tell you this), try http://:n/, where n is the port number. If the databases are to be served on the internet, your ISP or ICT staff member should give you a unique and static IP address to use for the FileMaker Server. Better still, a https domain name address should be provided. Type this address into your web browser. A web page will open and you should see a link to your visible database(s).

3. Click a link to a database to open it.

4. Type your username and password for the database. Click Login. It can take up to 20 seconds on a 1.6GHz single processor machine (much faster on machines with higher speed and with in-built dual or quad core processors) to display a reasonably complex database layout[5]. Fortunately SUNRISE Contacts 2017 is designed for simplicity in the design of the layouts to help render them more quickly in a web browser. Once the layout shows up in the web browser, it should be very quick to navigate through the records (and potentially through most other layouts given the consistency in layout design).

RECOMMENDATION
Turn off Firewall, or set up the Firewall such that a port can be opened exclusively for FileMaker Server to publish the layouts and data over a network.

---

5 It is also dependent on how fast the computer doing the serving (i.e., the one with FileMaker Server installed) is. So if you intend to serve FileMaker databases to others, use the fastest machine possible (preferably with SSD technology).

FileMaker Server PHP
Accessing your databases the geeky way

The alternative to WebDirect is to naturally go the geeky way, which is to apply PHP, XML or REST APIs as your sole means of communicating with FileMaker Server and your database.

We will leave the XML option aside as PHP is more popular. Whereas REST (or Representational State Transfer) is the new standard compliant technology introduced in FileMaker Server 16, but it isn't yet fully matured in this release.

Just for your information, REST will soon replace PHP and XML as the standard way to communicate with FileMaker Server and all other third-party services, such as Google Maps, Microsoft SharePoint, and so on. It uses HTTP methods of applying the four basic operations to any data (and records, and eventually scripts when the technology is more refined in FileMaker Server). You may have seen these HTTP methods in the web address bar through the words POST, GET, PUT and DELETE. Or if you are familiar with web forms, you may have seen in the HTML code the POST method as a means of transferring data in the form to a PHP text file handler that then decides where the data should be sent for processing. The GET method is the most common. This is regularly used in web pages to read (or get) data from somewhere and display it on the screen. With these four HTTP methods, you can do pretty much everything you need to Create (or POST), Read (or GET), Update (or PUT) and Delete (sometimes given the acronym CRUD by geeky people to help them remember) any kind of data, including the records themselves when it comes to FileMaker Server.

The version of REST currently introduced into FileMaker version 16 is limited to the POST and GET methods[6]. Furthermore, it will not allow scripts inside FileMaker to be run as yet using these HTTP methods. However, when REST gets its full set of HTTP APIs and ability to use them to handle various aspects of a FileMaker database, expect this to be the way of the future.

Until REST is fully integrated into the next release of FileMaker Server and can use all the HTTP methods available to its fullest extent, let us focus on PHP for now.

The PHP approach to accessing data in a FileMaker database served by FileMaker Server and the ability to perform two of the most important CRUD operations is actually simply than many people think. However, it does require you to dive into a little PHP and HTML programming to see what needs to be added for communication with FileMaker Server to be successful. Because of this situation, it is not surprising most non-geeky people will prefer the WebDirect approach. However, for those willing to try

---

6 FileMaker developers employ the HTTP methods using the *Insert from URL* script step. For a list of APIs the script step can access and grab information, visit https://www.programmableweb.com/apis/directory.

their hand with PHP and HTML, here are the essential details you need to know to make this work for you.

The essence of FileMaker PHP

Imagine you have a basic HTML text file written like so,

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>HTML Form</title>
</head>

<body>

<form action="message_handler.php" method="post">

<label>First Name:</label>
<input name="First_Name" type="text" /><br />

<label>Last Name:</label>
<input name="Last_Name" type="text" /><br />

<label>Email:</label>
<input name="email" type="text" /><br />

<label>Message:</label>
<textarea name="message" cols="40" rows="20"></textarea>

<input type="submit" value="Send" />

</form>

</body>
</html>
```

The critical part that actually is the HTML form is shown in blue. The rest of the text surrounding the form itself is what you need to tell a web server that this text file is written in HTML and should be rendered nicely on a web browser with the help of the HTML commands shown in the page. For the purposes of this example, let us save this text file as *myHTMLform.htm*.

As you can see, we have a web form with four fields together with corresponding labels telling you what they are. Where you see text between <label>…</label> means it will be shown in the web browser. Where you see <input … /> means an empty box (or field) will appear next to the label, which is the place where you can type text inside the box. The <input type=”submit”… /> simply means a button will be shown to help you click on it in order to submit the data in the fields to a PHP file

designed to handle the data called *message_handler.php*. We know it will be submitted to this file because the HTTP method chosen is called POST, much like the way a post office worker will post your envelope containing data to a specified address (in this case a PHP file named *message_handler.php*).

The above code is all good and fine if you merely want the PHP handler file to send the data in the form to some location for you to access and later re-type the data into a FileMaker record. But why would you want to do that? To save time, you can get the PHP handler file to go direct to the FileMaker database, create a new record, and store the data automatically with the help of FileMaker Server. Before we look at this PHP handler file more closely to see how this works, let us improve the versatility of the HTML web form.

Notice how the text fields in the HTML form appear empty when viewing in a web browser. When you enter data, there is a risk that you could lose the data if, for instance, you decide to refresh the page, or move to another page and then return to the HTML form. You might consider this to be okay. But what if you want to retain the data for a little while longer? In that case, let us introduce some useful improvements to the form as shown below in red.

```
<?PHP
/* Session start is essential to retain information even during a page
refresh */
session_start();

/* To retrieve data previously entered when the form was incomplete, we
create this function and use it in the HTML form.*/
function getField($field) {
if ( array_key_exists('post',$_SESSION) and
array_key_exists($field,$_SESSION['post']) ) {
        return $_SESSION['post'][$field];
  }
}

?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>HTML Form</title>
</head>

<body>

<form action="message_handler.php" method="post">

<label>First Name:</label>
```

```
<input name="First_Name" type="text" value="<?PHP echo
getField('First_Name');?>" /><br />

<label>Last Name:</label>
<input name="Last_Name" type="text" value="<?PHP echo
getField('Last_Name');?>" /><br />

<label>Email:</label>
<input name="email" type="text" value="<?PHP echo getField('Email');?>"
/><br />

<label>Message:</label>
<textarea name="message" cols="40" rows="20" value="<?PHP echo
getField('Message');?>"></textarea>

<input type="submit" value="Send" />

</form>

</body>
</html>
```

Notice how we remain consistent in the names of the text fields when introducing the PHP code to retrieve data. It is recommended that you continue this with the names of the fields used in the FileMaker database. Or else FileMaker Server will get confused later, and the PHP handler file will have to be programmed in a more complex way in order to create what we call the equivalent of the Rosetta Stone of PHP programming to link the fields names in your HTML form and PHP code to the names of the fields in the FileMaker database. Not hard to do, but it is best to keep things simple by being consistent in the naming of your fields, especially if you don't like to do much programming.

You will also need to change the file extension of .htm of the HTML form to .php. Why? You need to tell the web server that there is PHP code in the HTML form so that it can process code correctly. Otherwise the PHP code will appear in the web browser, and you probably don't want to do that. Changing the file extension to PHP means the PHP code will not be shown to users in a web browser. The code will be processed by the web server and converted to appropriate data on the screen at the right time if required, and/or sent to wherever you want the data to go for further processing.

Now comes the essence of sending data in the HTML form to the FileMaker database. Let us open a new text file, and type the following:

```
<?PHP

/* Require FileMaker's PHP APIs */
require_once('FileMaker.php');

/* Connect to the FileMaker database */
```

```
$fm = new FileMaker('[name of Filemaker database].fmp12','[IP address of FileMaker database]','[username]','[password]');

?>
```

Save this text file as *message_handler.php*. Place the PHP handler file in the same folder as *myHTMLform.php*. Otherwise you may end up having to write more code to tell the web server where to find the text files. Not good if you are not the geeky type and are somewhat allergic to programming.

As you can see from the text within the PHP handler file, we have introduced two critical pieces of code. These are considered essential to tell the web server that there is a FileMaker Server running in the background and there are specific APIs developed by FileMaker, Inc. to tell FileMaker Server what to do in a FileMaker database. Also, you need to tell FileMaker Server the details of what the FileMaker database is called, its location, and how to authenticate in order to make changes to data and records in the database.

It is clear all the hard work is done by FileMaker.php as installed by FileMaker Server. So there is little you need to do other than tell your main web server hosting your web site that it exists in your PHP handler file and should be used when communicating with FileMaker Server and your FileMaker database.

The other critical piece of code is merely to let FileMaker Server know which database it should be looking at and the necessary information it needs to access the data and records of the database.

The next step is not critical, but you may want to include the ability to validate the data in the HTML form within the PHP handler file. The most obvious validation you may wish to perform is to check for empty text fields. Of course, this is not essential for sending data to a FileMaker database. And the database itself can work this out with the right scripts set up and running in the background. However, to keep the performance of FileMaker Server and the database itself running optimally, you can do the checking directly in PHP. So let's add the following code (in red) to the PHP handler file:

```
<?PHP

*/ Check to see if the data received is empty */
if ( empty($_POST['First_Name']) or
     empty($_POST['Last_Name']) or
     empty($_POST['Email'])
     empty($_POST['Message']) ) {

/* If any of the HTML form text fields is empty... */

/* STEP 1: At the very least save the submitted information in case one or more fields did have some information typed into them */
```

```
$_SESSION['post']=$_POST;

/* STEP 2: Show an error message on the HTML web form. */
$_SESSION['errorMessage'] = 'Not all fields have been entered correctly.
Please supply the required information for all the fields(*).';

/* STEP 3: Send the user back to the HTML form to see the data entered
and the message of what happened. */
header("Location: myHTMLform.php");

}

else {

/* Everything looks okay... */

/* Clear the error message in case there is something there (perhaps from
a previous error) */
$_SESSION['errorMessage']='';

/* Now we are ready for the fun part */
/* Require FileMaker's PHP APIs */
require_once('FileMaker.php');

/* Connect to the FileMaker database */
    $fm = new FileMaker('[name of FileMaker database].fmp12','[IP address
of FileMaker database]','[username]','[password]');

}

?>
```

The validation section is done for you. If this looks a bit complicated, just copy and paste into your text file, and bear in mind the field names should be the same.

To complete the validation process, you should also add to the HTML form the following code (in red):

```
<?PHP
/* Session start is essential to retain information even during a page
refresh */
session_start();

/* To retrieve data previously entered when the form was incomplete, we
create this function and use it in the HTML form.*/
function getField($field) {
if ( array_key_exists('post',$_SESSION) and
array_key_exists($field,$_SESSION['post']) ) {
        return $_SESSION['post'][$field];
  }
}
```

```
?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>HTML Form</title>
</head>

<body>

<form action="message_handler.php" method="post">

<label>First Name:</label>
<input name="First_Name" type="text" value="<?PHP echo
getField('First_Name');?>" /><br />

<label>Last Name:</label>
<input name="Last_Name" type="text" value="<?PHP echo
getField('Last_Name');?>" /><br />

<label>Email:</label>
<input name="email" type="text" value="<?PHP echo getField('Email');?>"
/><br />

<label>Message:</label>
<textarea name="message" cols="40" rows="20" value="<?PHP echo
getField('Message');?>"></textarea>

<div style="color:red"><?PHP echo $_SESSION['errorMessage']; ?></div>
<input type="submit" value="Send" />

</form>

</body>
</html>
```

By doing so, any messages coming from the PHP handler file will appear on the HTML form.

The time has come to send data from the HTML form to the FileMaker database (yay!). With the name of the database, IP address, and authentication details already supplied, and we have the FileMaker APIs available, we can immediately send the data to the database using the command:

```
$newRecord=$fm->newAddCommand('[Layout Name in FileMaker
Database',$_POST);
```

where [Layout Name in FileMaker Database] must correspond to the exact

name of the layout in the database containing the fields of the exact same name as in the HTML form. If for any reason, the field names are not the same, you will have to use:

```
$newRecord=$fm->newAddCommand('[Layout Name in FileMaker Database]');

    $newRecord->setField('[Layout Field Name 1]', $_POST['First_First']);
    $newRecord->setField('[Layout Field Name 2]', $_POST['Last_Name']);
    $newRecord->setField('[Layout Field Name 3]', $_POST['Email']);
    $newRecord->setField('[Layout Field Name 4]', $_POST['Message']);
```

This is what is meant by creating the Rosetta Stone in PHP programming. You have to translate the field names in the HTML form to correspond to the field names in the FileMaker database so FileMaker Server knows what to do. Not too complicated, but for those less geeky among us, you can appreciate why keeping the field names consistent can save you time in typing more text in the PHP text file. Thus, the PHP handler file will look like this (assuming the field names are consistent):

```
<?PHP

*/ Your validation section */
if ( empty($_POST['First_Name']) or
     empty($_POST['Last_Name']) or
     empty($_POST['Email'])
     empty($_POST['Message']) ) {

/* If empty… */
$_SESSION['post']=$_POST;
$_SESSION['errorMessage'] = 'Not all fields have been entered correctly. Please supply the required information for all the fields(*).';
header("Location: myHTMLform.php");

}

else {

/* If okay… */
$_SESSION['errorMessage']='';

require_once('FileMaker.php');

$fm = new FileMaker('[name of FileMaker database].fmp12','[IP address of FileMaker database]','[username]','[password]');

/* Prepare FileMaker Server to create a new record and adding the data */
$newRecord=$fm->newAddCommand('[Layout Name in FileMaker Database',$_POST);

/* Tell FileMaker Server to execute what needs to be done */
$result = $newRecord->execute();
```

```
}

?>
```

Now we can stop there if we wish and be content with our newfound insights into FileMaker PHP. Or we can run a script in the FileMaker database as well. If you are feeling game enough to try, add this to your PHP handler file:

```
<?PHP

*/ Your validation section */
if ( empty($_POST['First_Name']) or
     empty($_POST['Last_Name']) or
     empty($_POST['Email'])
     empty($_POST['Message']) ) {

/* If empty… */
$_SESSION['post']=$_POST;
$_SESSION['errorMessage'] = 'Not all fields have been entered correctly.
Please supply the required information for all the fields(*).';
header("Location: myHTMLform.php");

}

else {

/* If okay… */
$_SESSION['errorMessage']='';

require_once('FileMaker.php');

$fm = new FileMaker('[name of FileMaker database].fmp12','[IP address of
FileMaker database]','[username]','[password]');

/* Prepare FileMaker Server to create a new record and adding the data */
$newRecord=$fm->newAddCommand('[Layout Name in FileMaker
Database',$_POST);

/* Run a script in FileMaker database */
$newRecord->setScript('[Script_Name]');

/* Tell FileMaker Server to execute what needs to be done */
$result = $newRecord->execute();

}

?>
```

where [Script_Name] should correspond to the exact name of the script in the FileMaker database that will run immediately after the new record has

been created and the data from the HTML form added to the FileMaker fields. This means you can design the script to send an email to the user to confirm the information has been received and any other processing to the record and data as you see fit.

Finally, $result can be useful if you need to show a message on the screen for the user as confirmation that everything is okay, or not. So add the final piece to the PHP handler file like so,

```
<?PHP

*/ Your validation section */
if ( empty($_POST['First_Name']) or
     empty($_POST['Last_Name']) or
     empty($_POST['Email'])
     empty($_POST['Message']) ) {

/* If empty… */
$_SESSION['post']=$_POST;
$_SESSION['errorMessage'] = 'Not all fields have been entered correctly.
Please supply the required information for all the fields(*).';
header("Location: myHTMLform.php");

}

else {

/* If okay… */
$_SESSION['errorMessage']='';

require_once('FileMaker.php');

$fm = new FileMaker('[name of FileMaker database].fmp12','[IP address of
FileMaker database]','[username]','[password]');

/* Prepare FileMaker Server to create a new record and adding the data */
$newRecord=$fm->newAddCommand('[Layout Name in FileMaker
Database',$_POST);

/* Run a script in FileMaker database */
$newRecord->setScript('[Script_Name]');

/* Tell FileMaker Server to execute what needs to be done */
$result = $newRecord->execute();

/* Show the success or otherwise of sending the information */
if (FileMaker::isError($result)){
$error = $result->getMessage();

/* Show error message */
?>
```

```
<h2>It seems the server is down or something else happened</h2>
<h4>Please send us an email to [email address], or try to re-submit
again.</h4>
<p>Error: <?PHP echo $error;?>.</p>

<?PHP
exit;
}

else{
/* No problems? Show thank you message */
?>

<h2>A confirmation email has been sent to you.</h2>
<h4>Thank you.</h4>

<?PHP
/* It is a good idea to clear out the saved data in the HTML form at this
stage for security reasons */
if(array_key_exists('post',$_SESSION)){ unset($_SESSION['post']); }
      exit;
   }

}

?>
```

That is the hardest part out of the way! You now have the essential knowledge of how to send data to a FileMaker database using PHP via FileMaker Server.

Which should I use?
The geeky or non-geeky way?

You have a choice of WebDirect or PHP to access your databases. The question is, which one should you use? For some people, the PHP method is considered the better way to go for internet use, only because you could have potentially hundreds, if not thousands, of anonymous users around the world accessing your web pages and possibly sending you information to store in your FileMaker database. The need for speed in server performance is a must. Therefore, a combination of PHP and HTML (together with a sparing use of other web technologies, such as Javascript) does help to speed up the process of delivering information to people's web browsers while retaining a sense of "prettiness" in the pages viewed (but don't overdo it too much with lots of pictures). Only use a database to store information when users are ready to submit a form for maximum performance. Alternatively, a well-designed and simple layout with changeable information delivered by a database can be rendered relatively fast through WebDirect and will still serve you well as a web site delivery mechanism.

In terms of security, some people feel safer not to have a complete and single database on a single server doing everything, including delivering and rendering with WebDirect your entire web site. So you might get recommended to go the PHP way. The advantages are that PHP can do a lot of checking of the data from HTML forms, and any data collected can be sent off to a FileMaker database when the check and balances are passed. And for greater security, the database can be hidden from a web browser. So long as the username and password used in a PHP script to access the FileMaker database are kept in a separate text file that is located on another server, what you have is a fairly secure way of delivering a service and gathering information on the internet. Then again, in WebDirect, you could choose to build two databases: the stripped back front-line database made visible (with auto-login) for public access to web information, and a more sophisticated "for office use" database in the background hidden from web users. The front-line database will have layouts that are the web pages of your web site, and any tables and fields to form the database component side of it can provide the changeable information relevant to the web site as well as store information from users. Whereas the other more sensitive database that has everything should at the very least be hidden from the internet and ideally located elsewhere away from the main web server. Let the main database import regularly the records in the table(s) of the front-line database. Indeed, if this siphoning off of records from the front-line database is done in a timed fashion, say, every 30 minutes, any hacker that can somehow get into the front-line database will not be able to see very much. At best he/she will see the web pages and its information, and at worst a few records of personal information from users collected over the 30-minute period. Little damage can be had through this method. And certainly the hacker will not be able to access the main database.

In conclusion, the choice of whether to go the WebDirect or the more geeky PHP/XML or REST approach will depend on the nature of your web site. If you expect many users to be regularly accessing data in your FileMaker database and visiting lots of complex web pages making up your web site, performance requirements will probably sway you towards the PHP approach[7]. In other situations, and given the simplicity in rendering layout designs in an accurate way for a modern web browser, you may find it much easier to deliver your web site and collect information from web forms with a single front-line database and let WebDirect do all the work. Then use another database to manage the front-line database and extract any records.

Securitywise, if you set up the databases and/or any PHP the right way, neither approach will be more secure. It will come down to the performance aspect that will ultimately dictate the approach you should take.

---

7 If you can create a simple web page design template that does not need to change for all the web pages, then rendering the web page with WebDirect can be potentially fast. And it will be even faster once the rendering is complete, since the only thing to change afterwards is the core information within the body of the HTML, the page title, and any title picture. Perhaps this is the key to delivering a high performance web site using WebDirect even if hundreds or thousands of anonymous users around the world are accessing the web site simultaneously.