AMFPHP & Flex *(created by Steve Stomp)*

This chapter discusses how to setup AMFPHP in order to work with Flex 3.0. In this tutorial I will be using AMFPHP 1.9 and Flex Builder 3.0. AMFPHP can be downloaded at http://sourceforge.net/project/showfiles.php?group_id=72483. A trial version of Flex Builder can be downloaded at http://www.adobe.com/products/flex.

AMF & AMFPHP

AMF stands for Action Message Format. It is a compact binary file that is used to represent a serialized ActionScript object. Flash Player uses AMF for data storage and data exchange. It was designed to serialize and deserialize quickly under low memory and slower CPU conditions, making it perfect for the Web. AMF data is parsed directly into objects, meaning there is no lag for interpreting or parsing AMF, making the creation of objects complete in a single pass (Arnold W., 2008). AMF 3, the default serialization for ActionScript 3, provides various advantages over AMF 0 (AS1 & AS2). AMF 3 supports sending int and uint objects as integers and supports data types that are available only in AS3, such as ByteArray, ArrayCollection and IExternalizable (Arnold W., 2008). AMF is much faster than XML when large amounts of data are involved. With XML the webserver receive text data, which needs to be parsed by the webserver, AMF uses binary data. The benefit is that the data is more compressed, and thus is faster send over the web. (see figure e.1) AMFPHP allows users to easily exchange data between a SWF file and PHP. The PHP methods can be called directly from ActionScript code. AMFPHP automatically converts ActionScript data types to PHP data types.

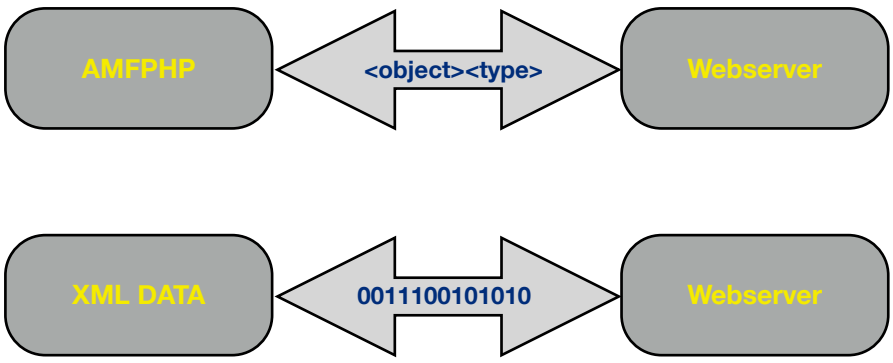


figure e.1, xml vs amfphp

Setup up

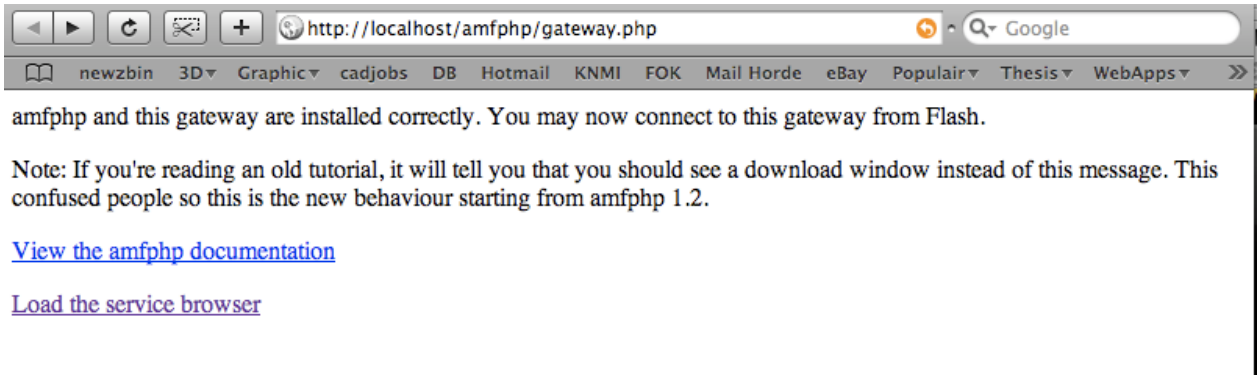
AMFPHP requires a web server with a PHP release newer than 5.2.3. If you don't have acces to web server you can easily install it on your system using WAMP (for Windows, www.wampserver.com) or MAMP (for Mac OS X, www.mamp.info). These application allow you to create web applications with Apache, PHP and the MySQL database. In this tutorial I use MAMP.

1. Once you downloaded AMFPHP you need to install it in your webroot folder.

    i. Windows > c:/wamp/

    ii. Mac OS X > [username]/sites

2. To test the server installation open your webbrowser and go to http://localhost/amfphp/gateway.php. If the installation is successfull, you should see the same image as in the figure below.



3. AMFPHP provides a browser for testing php services. To access it, go to http://localhost/amfphp/browser/ . The first time you access the browser it will display a configuration window. Here you can change your gateway location and encoding format.

Mail service

In this section I will explain how to create a simple php script to send mail. Furthermore I show you how to create ActionScript code in order to send variables to the webserver.

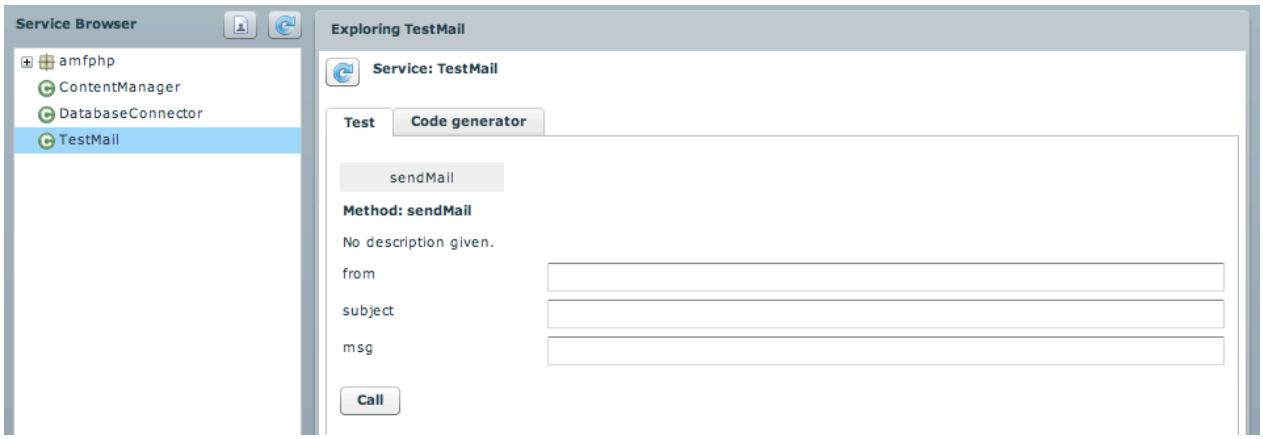
1. Open up a script editor to create a PHP script. This class will enable us to send mail through PHP.

```
<?php
class TestDB {
        function sendMail($from, $subject, $msg) {
                $emailTo = “yourname@mail.nl;
                $email_info .= "This is a demo\n\n";
                $email_info .= "----------------------";
                $email_info .="Afzender: ".$from."\n";
                $email_info .= "Bericht: ".$msg."\n";
                $mailheaders = "From: ".$from."\n";
                $mailheaders .= "Reply-To: ".$from."\n\n";
                if(mail($emailTo, $subject, $email_info, $mailheaders)) {
                        return true;
                } else {
                        return false;
                }
        }
}
?>
```

2. It is important that AMFPHP is able to locate this file. Save the PHP file in localhost/amfphp/services/. This services folder is the location for all your PHP scripts.

3. Open up the service browser.

4. If everything went correct you should see a figure looking like:



5. On the left side you see an overview of all your PHP classes located in the services folder. The section on the right displays the methods of a class.. In our mail method we expect 3 variables, see previous figure. By clicking on “Call” you run the script.

6. If you want to run this script over the internet, upload the whole AMFPHP folder to your webserver. Make sure not to upload the hidden file .htacces. With some providers hosting will become unavailable if this file is located in your web folder.

Combining AMFPHP, MySQL and ActionScript 3

Here I will explain how to communicate with a database through the use of AMFPHP and AS3. First we will create a simple database in MySQL and will use AMFPHP to retrieve and alter content from this database. This content will be converted to AS3 variables in order to do operations with it. In this first tutorial I will be only displaying this content in AS3 text field. In the second tutorial I will explain how to creating a login component using AMFPHP.

Getting data out of MySQL

1. Open up your phpMyAdmin and create a new database. For example, this SQL script

```
SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";
--
-- Database: `vuDB`
-- --------------------------------------------------------
--
-- Table structure for table `introAMF`
--

CREATE TABLE `introAMF` (
`id` int(11) NOT NULL auto_increment,
`username` text,
`password` text,
`fname` text,
`lname` text,
PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 AUTO_INCREMENT=1 ;
```

2. Fill the database with data. For example;

```
INSERT INTO `vuDB`.`introAMF` (
```

```
`id` ,
`username` ,
`password` ,
`fname` ,
`lname`
)
VALUES (
'1', 'login', 'admin', 'John', 'Smith'
), (
NULL , 'pocket', 'rockets', 'Jane', 'Doe'
);
```

3. In order to communicate with the database, the PHP script has to make connection with it. I use a class called “DatabaseConnector” that handles this. At the top of this script you can alter the database properties. I am running this from my local webserver (**localhost**), my database is called **vuDB** and my login and pass are **root**. Save this file in the services folder.

```
<?php
class DatabaseConnector
{
        private $host           = "localhost";
        private $dbName         = "vuDB";
        private $user           = "root";
        private $pass           = "root";

        private $link    = null; //to store db link resource id

        function DatabaseConnector() {
        //empty because all private methods are already defined
        }
        private function connect() {//executed when no connection is found, variables only visible in the class
                $this->link = mysql_connect($this->host, $this->user, $this->pass);
                mysql_select_db($this->dbName, $this->link);
        }
        /**
        * Make database connection
        * @acces private
        * @returns database link, to be used in other sql calls
        */
        public function getConnection()        {
                if($this->link == null) {
                        $this->connect();
                }
                return $this->link;
        }
}
?>
```

4. Now we need to create a PHP function to retrieve the data. Right after <?php expand TestDB.php with require_once ‘DatabaseConnector.php’; and extend the class with DatabaseConnector.

```
<?php
require_once ‘DatabaseConnector.php’;
```

```
class TestDB extends DatabaseConnector {
        ---
```

5. Creating function to communicate with the database are very easy now. First we are going to create a variable that makes a connection to the database:
   $link = $this->getConnection();

6. Then we create a query:
   $query = sprintf("SELECT * FROM introAMF");

7. Then we execute the query:
   $result = mysql_query($query, $link);

8. And finally it need to be returned:
   return $result;

9. The whole function will look like this:

```
function retrieveAll() {
        $link = $this->getConnection();
        $query = sprintf("SELECT * FROM introAMF");
        $result = mysql_query($query, $link);
        return $result;
}
```

10. Now open the AMFPHP service browser and test the script. Your output has to look something like the figure below.

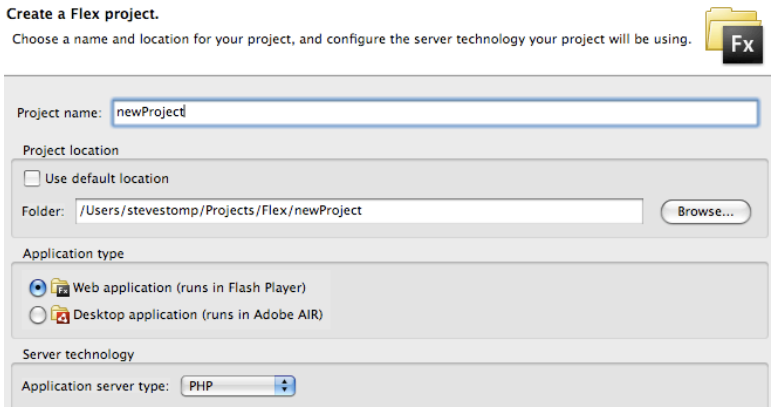
Info | Results | Tree view | RecordSet view | Trace

id	username	password	fname	lname
1	login	admin	John	Smith
2	pocket	rockets	Jane	Doe

Bringing MySQL data in a AS3 Datagrid

In this section we are going to create a AS3 environment which will be used to perform operation on the database.

1. Open Flex Builder and create a new project. Make sure Application server type is set to PHP.



2. Add a Datagrid in the mxml file to represent the data. In order to fill the Datagrid with the data out of the database. Make sure the dataField property is set to corresponding name in the MySQL database.

```
<mx:DataGrid x="502" y="86" id="userDB">

      <mx:columns>
            <mx:DataGridColumn headerText="Username" dataField="username"/>
            <mx:DataGridColumn headerText="Password" dataField="password"/>
            <mx:DataGridColumn headerText="First Name" dataField="fname"/>
            <mx:DataGridColumn headerText="Last Name" dataField="lname"/>
      </mx:columns>
</mx:DataGrid>
```

3. Now create a Form. This form will be used to add new data to the database.

```
<mx:Form x="199.5" y="86" width="283" backgroundColor="#000000" cornerRadius="20"
      borderStyle="solid">

      <mx:FormItem label="login:" width="100%" color="#FFFFFF" fontSize="12">
            <mx:TextInput id="userField" width="100%" color="#000000"/>
      </mx:FormItem>
      <mx:FormItem label="password:" width="100%" color="#FFFFFF" fontSize="12">
            <mx:TextInput id="passwordField" width="100%" color="#000000"/>
      </mx:FormItem>
      <mx:FormItem label="first name:" width="100%" color="#FFFFFF" fontSize="12">
            <mx:TextInput id="fnameField" width="100%" color="#000000"/>
      </mx:FormItem>
      <mx:FormItem label="last name:" width="100%" color="#FFFFFF" fontSize="12">
            <mx:TextInput id="lnameField" width="100%" color="#000000"/>
      </mx:FormItem>
</mx:Form>
```

4. Create two buttons to execute the operation.
5. Now we are going to add ActionScript code. In a mxml file ActionScript is written between <mx:Script> tag.
6. First create three variables:

   I. private var users:Array; (this variable must be Bindable)

   II. private var gateway:NetConnection;

   III. private var gatewayURL:string = “http://[your amfphp path]

7. The Array variable is going to be used to fill the Datagrid. The NetConnection variable enables us to make a connection to the database. This class creates bidirectional connection between a Flash Player and web server. A Netconnection object is like a pipe between the client and the server.

8. Now add a method to make the connection with the web server

```
private function setConnection() : void {

      gateway = new NetConnection;
      gateway.connect(gatewayURL);
}
```

9. Create a method fillDG()
10. Add ‘setConnection();’ to this method
11. Create a variable of the class Responder. The Responder class provides an object that is used in NetConnection.call() to handle return values from the server related to the success or failure of specific operations. The first parameter in the Responder object is executed when the gateway call was succesfull and the section when the call is failed.

```
var responder:Responder = new Responder(returnDGData, returnFault);
```

12. Finally add the piece of code that will call the a method out of the PHP script.

```
gateway.call("TestDB.retrieveArray", responder);
```

13. The call operation of a Netconnection object accepts the following parameters:
    1. String command for selection of a method out of a PHP class
    2. Responder object for executions of a succesfull or failed call.
    3. Parameters that have to be sent to the PHP method (only if needed).

14. Now we are going to create the methods when for a succesfull and failed call. When a succesfull call occurs a object will be returned. The retreiveArray method will return an array object. The array object will be used to fill the datagrid object (userDB).

```
private function returnDGData(result:Array) : void {

      userDB.dataProvider = result;
}
```

15. If a fault occurs we like to show a error message. In this tutorial I choose for a popup. At the top of the script section add

```
import mx.controls.Alert;
```

15. The fault method looks like:

```
private function returnFault(obj:Object) : void {

      Alert.show("Error has accured");
}
```

16. Open TestDB.php and add the following code:

```
function retrieveArray() {

      $rows = array();

      $link = $this->getConnection();

      $query = sprintf("SELECT * FROM introAMF");

      $result = mysql_query($query, $link);

      while($row = mysql_fetch_array($result, MYSQL_ASSOC)) {

            array_push($rows, $row);

      }

      return $rows;

}
```

17. Finally we need to make sure the datagrd is filled when the application is started. Add creationComplete=”fillDG()” to the Application tag.
18. Run the application
19. Now we are going to create two methods in AS3. One to add data to the database and one to delete data from the database.

```
private function addNew() : void {
      setConnection();
      var responder:Responder = new Responder(returnNew, returnFault);
      gateway.call("TestDB.addNew", responder, userField.text, passwordField.text,
            fnameField.text, lnameField.text);
}
private function returnNew(obj:Object) : void {
      Alert.show("New item added to database");
      fillDG();
}
private function removeContent() : void {

      setConnection();
      var responder:Responder = new Responder(refreshList, returnFault);
      gateway.call("TestDB.removeUser", responder, userField.text);
}
private function refreshList(obj:Object) : void {
```

```
            Alert.show("Data is removed");
            fillDG();
        }
```

20. The AS3 methods both call a method in PHP class, addNew and removeUser. So add these methods in your PHP class.
21. The add method:

```
function addNew($login, $pass, $fname, $lname) {

        $link = $this->getConnection();

        $query = sprintf("INSERT INTO introAMF(username, password, fname, lname) VALUES ('".$login."', '".
$pass."', '".$fname."', '".$lname."')");

        $result = mysql_query($query, $link);

        return $result;

}
```

22. The remove method:

```
function removeUser($user) {

        $link = $this->getConnection();

        $query = sprintf("DELETE FROM introAMF WHERE username='".$user."'");

        $result = mysql_query($query, $link);

        return $result;

}
```

23. Assign the methods to the created buttons, by adding click="[method name]" to button tags. For example;

```
<mx:Button x="350" y="246" label="delete" click="removeContent()"/>
```

User Authentication

In the last tutorial I will show how to create a login component using AMFPHP and AS3

1. Create a new folder in your source folder. For example; *src>comps*

2. In that folder create a new mxml compontent. For example; *Login.mxml*

3. Select the container type you prefer. My login component is a VBox.

4. Create a form with a login input field and password input field. Don't forget to give both components an id.

5. I also add a text field to report the progress to the user.

6. And finally add a button. My mxml code looks like this.

```
<mx:VBox verticalGap="0" id="vbox1">

    <mx:Panel
        title="Login" id="loginPanel"
        horizontalScrollPolicy="off" verticalScrollPolicy="off"
backgroundAlpha="0.3">
    <mx:Form id="loginForm">
        <mx:FormItem label="Gebruikersnaam:" required="true" id="formitem1">
            <mx:TextInput id="loginCol"/>
        </mx:FormItem>
        <mx:FormItem label="Wachtwoord:" required="true" id="formitem2">
            <mx:TextInput id="passwordCol" displayAsPassword="true"/>
        </mx:FormItem>
    </mx:Form>
    <mx:VBox width="100%" id="hbox1" horizontalAlign="center">
      <mx:Text id="alertTxt" color="#FF0909" fontSize="11" selectable="false"
width="100%" fontWeight="bold"/>
    </mx:VBox>
```

```
        <mx:ControlBar id="controlbar1">
            <mx:Spacer width="100%" id="spacer1"/>
            <mx:Button label="Login" id="loginButton"
                enabled="{(loginCol.text.length == 0 || passwordCol.text.length == 0) ?
false : true}"
                useHandCursor="true" buttonMode="true"/>
        </mx:ControlBar>
        </mx:Panel>
    </mx:VBox>
```

7. Now add the ActionScript code. First add a <mx:Script> tag and import mx.controls.Alert.
8. This piece code has the same structure as the main application. So add the setConnection method and declare the required variables.
9. Now add a method to handle the click event

```
public function doLogin(event:Event) : void {

      trace('CLICKED');
      loginButton.addEventListener(MouseEvent.CLICK, loginHandler);
}
```

10. The doLogin method calls the loginHandler method. This method will connect to the webserver and calls the login method out of the PHP class. This class returns a true boolean value if a user exists in the database. If not, false will be returned.

```
private function loginHandler(event:MouseEvent) : void {

      trace('Do Login');
      alertTxt.text = "Een ogenblik alsjeblieft. Bezig met laden.";
      var responder:Responder = new Responder(loginRespHandler, returnFault);
      setConnection();
      gateway.call("TestDB.login", responder, loginCol.text, passwordCol.text);
}
private function returnFault(obj:Object) : void {
      Alert.show("Error has accured");
}
```

11. And the PHP method looks like:

```
function login($user, $pass)   {

      $link = $this->getConnection();

      $loggedIn = false;

      $query = sprintf("SELECT * FROM introAMF WHERE username='%s' AND

            password='%s'", $user, $pass);

      $result = mysql_query($query, $link);

      if(mysql_num_rows($result) > 0) {

            $loggedIn = true;

      }

      return array('response' => $loggedIn);

}
```

12. The Responder variable calls the loginRespHandler. If the response out of PHP is true the state of the login component will change to ‘succes’ (see code snippet below), and if false a text will appear saying that the values are incorrect. This variable will be called in the parent application. So make sure it is set to public.

```
<mx:states>

      <mx:State name="succes">
            <mx:RemoveChild target="{loginForm}"/>
            <mx:RemoveChild target="{alertTxt}"/>
            <mx:AddChild relativeTo="{hbox1}" position="lastChild">
                  <mx:TextArea height="64" fontFamily="Arial" fontSize="12">
```

```
                        <mx:text>Je login was succesvol. Klik op de button om verder
te gaan</mx:text>
                    </mx:TextArea>
                </mx:AddChild>
                <mx:SetProperty target="{loginButton}" name="label" value="Ga verder"/>
            </mx:State>
        </mx:states>
```

13. Now add the login component to the main application. In the Application tag add: xmlns:lg=”comps.*”.

    Right after the closing script tag add:

```
<lg:Login id="loginPanel" x="171" y="308" click="handleLogin(event)"/>
```

14. Add the handleLogin method to the main application

```
private function handleLogin(event:Event) : void {

      loginPanel.doLogin(event);
      if(loginPanel.valid) {
      currentState="loggedIn";
}
```

15. Finally add the ‘loggedIn’ state to the main application

```
<mx:states>

      <mx:State name="loginScreen">
            <mx:SetProperty target="{loginPanel}" name="x"/>
            <mx:SetProperty target="{loginPanel}" name="y"/>
            <mx:RemoveChild target="{form1}"/>
            <mx:RemoveChild target="{button1}"/>
            <mx:RemoveChild target="{button2}"/>
            <mx:RemoveChild target="{userDB}"/>
            <mx:SetStyle target="{loginPanel}" name="horizontalCenter" value="0"/>
            <mx:SetStyle target="{loginPanel}" name="verticalCenter" value="0"/>
      </mx:State>
      <mx:State name="loggedIn">
            <mx:RemoveChild target="{loginPanel}"/>
      </mx:State>
</mx:states>
```

16. Test your application.