

Creating an Email Form in Flash MX

SorBose, Inc. All Rights Reserved. www.sorbose.com

Flash cannot send email in the same way that a browser cannot send email. Let's take the model of a HTML form, dissect it and see how Flash is similar to it in many ways. We don't process forms in HTML. We design them, or rather looking at some of the forms, we add them onto html pages. Making any form in HTML relies on the use of the <form> tag and its various elements such as:

- <input type="text" name="userName">
- <input type="hidden" name="hasCookie">
- <input type="submit">

You then typically have a form action which points to a script/program that processes this form. This program is invoked normally when the user of the html form presses the **SUBMIT** button. Once that button is pressed, the browser packages everything in the <form> tag and sends it to the script/program that the form action points to. Browsers generally package this data in either of the two ways

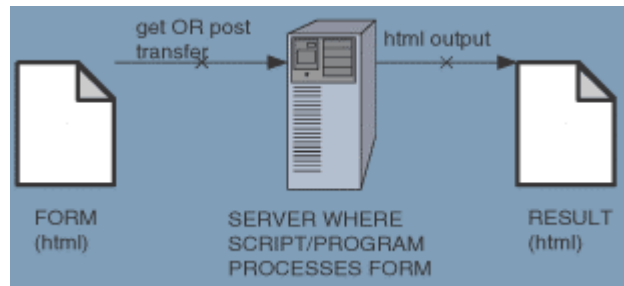
- GET
- POST

You as the developer specify along with other items in the form. Each input type specified in your form tag also has a name. This name is used by your program as the **variable name** with which to access the data. The browser's job is done when it has packaged the data and sent it to the url to which the form was pointing to. REMEMBER THIS! We will recall this paragraph later again

THE FORM ACTION

The script/program to which the HTML form has sent the data will then receive it and get to work. This script/program is no longer HTML but any of the various server-side languages such as PERL, PHP, Cold Fusion, ASP, Java, Server-side Javascript and numerous others. You take your pick and choose whatever you have access to. We'll use PHP in this tutorial. The purpose of this script/program could be to send email or book you a flight on an airline as an example. It is that program which performs the application logic and could be doing any number of things with the data. We as Flash Actionscript developers don't need to know how it does all those wonderful things. We just want it to do it and send us an **okay message**. That brings me to the other very important thing we need to remember.

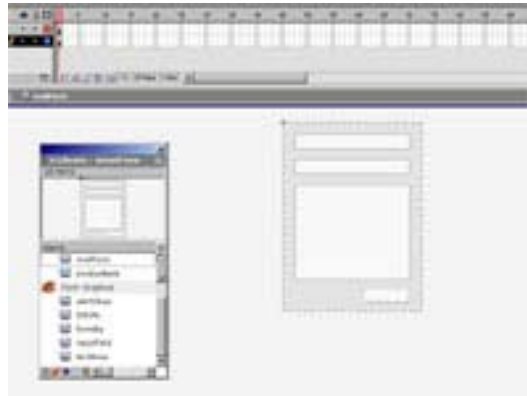
Typically, since the web is HTML based, this **okay message** comes as HTML code. The wonderful magic script/program which does all nice things to the data prints out some HTML which is **sent back** to the browser. The browser then shows it to us. The reason our program prints out HTML is for the browser to understand. It is also possible that our program found something wrong with the data and instead of an okay message it sends an **Error Message** in HTML. This time the browser will show you the error message. The thing to understand here is that this server program/script is doing the communication not the html code. Here is the basic model:



Flash is "almost" exactly the same. It replaces the browser yet provides the same purpose as the browser does. It displays whatever its been asked to display by the **Other** server program. So in our case our program tells Flash whether it sent out the mail or not, but and this is **really important**, the program should **NOT** send HTML. Why, because HTML is not understood by Flash. Formmail.pl script did not work because even though Perl was getting the variables via Flash, it was sending out messages back to Flash in a language that Flash does not understand. We need to send back messages in Flash's native language. Flash understands simple plain text. Just send some variable=value pairs such as sent=1 or sent=0 and Flash will understand that sent **1 means YES** and sent **0 means NO**. Flash likes plain text, no HTML. Okay then, so let's write a form in Flash.

Flash Form Creation

Let's design a form first. Here is a final screen shot. Design it anyway you like.

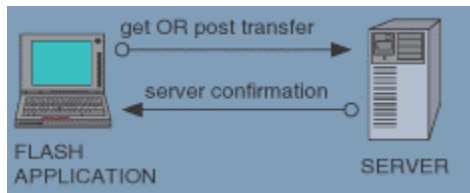


Now that we've made a library item appropriately called "mailForm", we need to write some actionscript to make the form functional. Things to do to make the form talk to the server:

- Variable Names to assign data
- Package Data and send to server most likely on an event (i.e. onRelease of submit button, but it can be anything)
- Write a method to receive a confirmation from the server
- Provide a server address for our mail program/script

Flash is better than a browser because unlike a browser, Flash is not stateless. It remembers what you've asked of it and what it was doing before you made the request. Simply put, Flash has a robust inbuilt language called Actionscript which allows some highly advanced options and control over the display and handling of data.

So a Flash-Server model looks more like this:



Even though there are 2 separate processes involved, just like HTML, the same application is still active even after sending out the data to the server and receiving a confirmation from the server. In that sense Flash is a smart tool with memory and that is why its an injustice to Flash, to think of it on the same lines as a stateless web page. Flash is more because it can act on data. This does not change the fact that the protocol is still exactly the same as in HTML. So finally let's get to some actionscript Variable Names:

```

/*The elements in the form
defined as an array of objects with
the following structure:

```

1. title : The title of the form input field as displayed to the user
2. type: The type of field made to let Flash know what it is dealing with
3. name : MUCHO IMPORTANTE' The variable name used by Flash to package data and send to server. This same name will be used by the server to access the data
4. parent : The movieclip instance which will house the dynamically created textField where the form end-user can type input.

```

*/
elements = new Array();
elements.push({title:"Name", type:"input", name:"name", parent:formMc.nameMc});
elements.push({title:"Email", type:"input", name:"email", parent:formMc.emailMc});
elements.push({title:"Message", type:"textArea", name:"message", parent:formMc.messageMc});
elements.push({title:"Send", type:"submit", name:"title", parent:formMc.submitMc});

```

```

/*We will create the textFields on the fly in a loop and add one more property to
each object in the array above called field. This field will contain a reference to the
textField so that we access its text later with ease.*/

```

Packaging Data

```

//Make a loadVars object to send to our PHP script
mailData = new LoadVars();

```

```

//Now get the variables to be sent to the PHP script
//These variables come from the form elements we designed earlier
//Scan through all the elements and populate mailData for every
//userInput type of field.. ie. input or textArea.. this list could include
//even more options such as checkBox, radioButton etc.
var i = elements.length;
while (i-->0) {
    switch (this.elements[i].type) {
        case "input":
        case "textArea":
            //Use same name as given to us in the form design
            //So PHP will also get the variable with the same name
            //For eg: if the name in our form is email.. then php will get
            //a variable called $email
            //This process is similar to HTML forms where your field name
            //becomes the variable name for your PHP script
            //NOTE: we escape the text to urlencode it for safe transfer
            //We will urldecode it in PHP

```

```

//Remember.. we made field property as shown in the entire
//code later
mailData[elements[i].name] = escape(elements[i].field.text);

/*Now when we are submitting data we don't want the user
to be able to edit the form in anyway.. we need to take away that ability
for the time being*/
elements[i].field.type = "dynamic"; //Don't allow editing
elements[i].field.selectable = false; //Don't allow editing

break;
} //End switch
} //End while loop

```

onLoad() i.e. upon confirmation

```

/*Things to do when data loads up from the server
ie. a confirmation is recieved*/
mailData.onLoad = function(){
    /*We are expecting 2 variable value pairs
    from the server

    1. status : A string with a message like Message Sent\nThank You
    2. sent : A number which is either 0 or 1. 0 meaning not sent and
    1 meaning sent sucessfully*/

    //Since all external data comes in as a string.. we convert
    //the sent variable from a string to a number.
    this.sent = Number(this.sent);

    if (this.sent == 1) {
        trace("Message Sent successfully");
    } else {
        trace("Message could not be sent");
    } //End if

    //also trace the other variable with a PHP message
    trace(this.status);
}; //End function

```

Send 2 Server

```

/*Provide the address for the script/program where Flash will send
information. This process is exactly like the <form action="script.cgi">
method.*/
var mailServer = "sendMail.php";

//We invoke the sendAndLoad method, new to Flash MX
//This will send everything inside the mailData object
//We packaged that object in the script above and we specify
// a target where to recieve the load. That object also happens
//to be mailData but could have been whatever we choose
//The final argument uses POST as the send method since
//POST is more reliable than GET for a number of reasons
mailData.sendAndLoad(mailServer, mailData, "POST");

```

These are key methods of data exchange between Flash and server. The final actionscript is more elaborate and can be seen below:

```

//Just a simple method to convert RGB values
//to their equivalent HEX numbers

```

```

//Used in this script for setting colors of userInput text
Math.rgb2Hex = function(r, g, b){
    return ((r << 16) | (g << 8)) | b;
};//End function

//Converts plainText 2 Html
String.text2Html = function(str, fontSize, fontColor, leading, align, fontFace) {
    if (fontFace == null) fontFace = "Verdana, Arial, Helvetica, sans-serif";
    if (fontSize == null) fontSize = 10;
    if (leading == null) leading = 1;
    if (align == null) align = "left";
    if (fontColor == null) fontColor = "#333333";

    //The return string
    var r = "<textformat leading=\""+leading+"\">";
    r+= "<p align=\""+align+"\">";
    r+="<font face=\""+fontFace+"\"";
    r+=" size=\""+fontSize+"\"";
    r+=" color=\""+fontColor+"\">";
    r+=str;
    r+="</font></p></textformat>";
    return r;
};//End function

//The mailForm

//save stage width and height
_global.stageD = {w:300, h:350};

MailForm = new Object();

//Makes the mailForm
MailForm.init = function(){
    //Local vars used by this function
    var i, mc, parent, parentSize, tFSize, tFormat, htmlTxt;

    //the the form to stage and position it in the center..
    this.formMc = attachMovie("mailForm", "mailFormMc", 1);

    //Center the form on stage
    this.formMc._x = (stageD.w - this.formMc._width)/2; //Center
    this.formMc._y = stageD.h - this.formMc._height - 2; //Bottom

    //Now design the elements in this form as an array of objects for easy access
    this.elements = new Array();
    this.elements.push({title:"Name", type:"input", name:"name",
parent:this.formMc.nameMc});
    this.elements.push({title:"Email", type:"input", name:"email",
parent:this.formMc.emailMc});
    this.elements.push({title:"Message", type:"textArea", name:"message",
parent:this.formMc.messageMc});
    this.elements.push({title:"Send", type:"submit", name:"title",
parent:this.formMc.submitMc});

    //The format to use for formatting the fields
    tFormat = new TextFormat();
    tFormat.font = "_sans";
    tFormat.size = 12;
    tFormat.color = Math.rgb2Hex(51, 51, 51);

```

```

tFSize = new Object(); //init size object for text fields

//How many elements did we get?
i = this.elements.length;
//Now make the fields in the elements
while (i--) {
    //Find out what type of field to make
    switch (this.elements[i].type){

        case "input": //Input field OR
        case "textArea": //textArea both have very similar actions

            //Who is the parent mc of this field
            parent = this.elements[i].parent;
            //What is the size of the parent of this element
            parentSize = {w:parent._width, h:parent._height};

            //Determine size of the title field and make it
            tFSize.x = 0;
            tFSize.y = -16;
            tFSize.w = Math.floor(parentSize.w);
            tFSize.h = Math.abs(tFSize.y);
            parent.createTextField("title", 1, tFSize.x, tFSize.y,
tFSize.w, tFSize.h);

            parent.title.selectable = false;
            parent.title.html = true;
            parent.title.htmlText =
String.text2Html(this.elements[i].title, 10, "#333333", 0, "left");

            //Determine size of the input field and make it
            tFSize.x = 2;
            tFSize.y = 2;
            tFSize.w = Math.floor(parentSize.w - (tFSize.x*2));
            tFSize.h = Math.floor(parentSize.h - (tFSize.y*2));
            parent.createTextField(this.elements[i].name, 2, tFSize.x,
tFSize.y, tFSize.w, tFSize.h);

            //Make the field
            //save a reference to the userInput field
            //this reference will make it easier to access the userInput
            //into this field when submitting data to the server
            this.elements[i].field = parent[this.elements[i].name];

            this.elements[i].field.type = "input";
            //only if we have text area..then set it to multiline true as
well

            if (this.elements[i].type == "textArea") {
                this.elements[i].field.multiline = true;
                this.elements[i].field.wordWrap = true;
            }//End if
            this.elements[i].field.setNewTextFormat(tFormat);

            break;

        case "submit": //Submit button
            //Who is the parent mc of this field
            parent = this.elements[i].parent;
            //What is the size of the parent of this element
            parentSize = {w:parent._width, h:parent._height};

```

```

        //Determine size of the title field and make it
        tFSize.x = 2;
        tFSize.y = 2;
        tFSize.w = Math.floor(parentSize.w - (tFSize.x*2));
        tFSize.h = Math.floor(parentSize.h - (tFSize.y*2));
        parent.createTextField("title", 1, tFSize.x, tFSize.y,
tFSize.w, tFSize.h);

        parent.title.selectable = false;
        parent.title.html = true;
        parent.title.htmlText =
String.text2Html(this.elements[i].title, 10, "#333333", 0, "center");

        //Add actions for the submit button
        parent.controller = this; //Reference to controller object
        parent.title.orginalText = this.elements[i].title; //Save
original text to allow color changes to it later
        parent.onPress = function(){
            //Give impression of press
            this.nextFrame();
            this.title.htmlText =
String.text2Html(this.title.orginalText, 10, "#FF6600", 0, "center");
        };//End function

        parent.onRelease = function(){
            this.prevFrame();
            this.title.htmlText =
String.text2Html(this.title.orginalText, 10, "#333333", 0, "center");
            //Send a message to the controller that we would like
to submit
            this.controller.doSubmit();
        };//End function

        parent.onReleaseOutside = function(){
            this.prevFrame();
            this.title.htmlText =
String.text2Html(this.title.orginalText, 10, "#333333", 0, "center");
        };//End function

        parent.useHandCursor = true;
        parent.hitArea = parent;

        break;
    }//End switch
} //End while loop

/*#####*/
//Bring in the Productbeta logo on the top right corner
//and link to www.productbeta.com
var logoMc = attachMovie("pbLogo", "pgLogo", 2);
logoMc._x = this.formMc._x + this.formMc._width - logoMc._width;
logoMc._y = 3;

logoMc.onRelease = function(){
    getURL("http://www.productbeta.com/", "_blank");
};//End function

/*#####*/

//Finally add a nice touch to the form..

```

```

        this.focusFirstField();
    }; //End function

MailForm.focusFirstField = function(){
    //automatically set selection to blink on the first
    //input type or textArea type element in our form design
    var tElements, i;
    tElements = this.elements.length;
    for (i=0; i<tElements; i++) {
        if (this.elements[i].type == "input" || this.elements[i].type ==
"TextArea") {
            Selection.setFocus(this.elements[i].field);
            break;
        } //End if
    } //End for loop
}; //End function

//Submits to the form to the server for processing
MailForm.doSubmit = function(){

    //Cannot submit if we are busy talking to server
    if (this.talking2Server) return;

    //Cover the stage with an alert box
    this.makeAlertBox();

    //Make a loadVars object to send to our PHP script
    this.mailData = new LoadVars();

    //Now get the variables to be sent to the PHP script
    //These variables come from the form elements we designed earlier
    //Scan through all the elements and populate mailData for every
    //userInput type of field.. ie. input or textArea.. this list could include
    //even more options such as checkBox, radioButton etc.
    var i = this.elements.length;
    while (i--) {
        switch (this.elements[i].type) {
            case "input":
            case "textArea":
                //Use same name as given to us in the form design
                //So PHP will also get the variable with the same name
                //For eg: if the name in our form is email.. the php will get
                //a variable called $email
                //This process is similar to HTML forms where your field name
                //becomes the variable name for your PHP script
                //NOTE: we escape the text to urlencode it for safe transfer
                //We will urldecode it in PHP
                this.mailData[this.elements[i].name] =
escape(this.elements[i].field.text);

                /*Now when we are submitting data we don't want the user
                to be able to edit the form in anyway.. we need to take away
                that ability
                for the time being*/
                this.elements[i].field.type = "dynamic"; //Don't allow
                editing
                this.elements[i].field.selectable = false; //Don't allow
                editing

                break;
        } //End switch
    }
};

```

```

} //End while loop

//Save a controller reference to work with mailForm from inside mailData
this.mailData.controller = this;

this.mailData.onLoad = function(){
    //Send yourself to the controller
    //so it can process and decide what to do with you
    this.controller.onMailResult(this);
}; //End function

//Okay we are ready to send data to the server.
//Flag this process to let Flash know that we are talking to the server
this.talking2Server = true;

//Let's talk!
//var mailServer =
"http://www.productbeta.com/tutorials/flash_email/scripts/sendMail.php";
var mailServer = "/tutorials/flash_email/scripts/sendMail.php";
this.mailData.sendAndLoad(mailServer, this.mailData, "POST");
}; //End function

MailForm.makeAlertBox = function(){
    this.formMc.formBg.duplicateMovieClip("alertCover", 1);
    this.formMc.alertCover.attachMovie("alertBox", "alertBox", 1);

    //Center the alert box
    this.formMc.alertCover.alertBox._x = (this.formMc.alertCover._width -
this.formMc.alertCover.alertBox._width)/2;
    this.formMc.alertCover.alertBox._y = (this.formMc.alertCover._height -
this.formMc.alertCover.alertBox._height)/2 - 20;

    //Make a text field in there to display PHP script results
    //Note size of the field is hard-coded simply because it does not matter
    //in this case since our alert box is pre-designed and these numbers
    //are simply an extension of that design
    this.formMc.alertCover.alertBox.createTextField("phpMsg", 1, 5, 25, 185, 52);
    this.formMc.alertCover.alertBox.phpMsg.selectable = false;
    this.formMc.alertCover.alertBox.phpMsg.wordWrap = true;
    this.formMc.alertCover.alertBox.phpMsg.multiline = true;
    this.formMc.alertCover.alertBox.phpMsg.html = true;
    this.formMc.alertCover.alertBox.phpMsg.htmlText = String.text2Html("Sending
your Message...\nPlease Wait", 10, "#333333", 2, "left");

    //Now make a text field for the okay button
    var tFSize = new Object();
    tFSize.x = 1;
    tFSize.y = 1;
    tFSize.w = this.formMc.alertCover.alertBox.okayMc._width - (tFSize.x*2);
    tFSize.h = this.formMc.alertCover.alertBox.okayMc._height - (tFSize.y*2);
    this.formMc.alertCover.alertBox.okayMc.createTextField("title", 1, tFSize.x,
tFSize.y, tFSize.w, tFSize.h);
    this.formMc.alertCover.alertBox.okayMc.title.selectable = false;
    this.formMc.alertCover.alertBox.okayMc.title.html = true;
    this.formMc.alertCover.alertBox.okayMc.title.htmlText =
String.text2Html("OK", 9, "#333333", 0, "center");

    this.formMc.alertCover.alertBox.okayMc.controller = this;
    this.formMc.alertCover.alertBox.okayMc.onPress = function(){
        this.nextFrame();
    };
};

```

```

                this.title.htmlText = String.text2Html("OK", 9, "#FF6600", 0,
"center");
            };//End function

            this.formMc.alertCover.alertBox.okayMc.onRelease = function(){
                this.prevFrame();
                this.controller.closeAlert();
            };//End function

            this.formMc.alertCover.alertBox.okayMc.onReleaseOutside = function(){
                this.prevFrame();
                this.title.htmlText = String.text2Html("OK", 9, "#333333", 0,
"center");
            };//End function
            //Hide the okay button for now
            this.formMc.alertCover.alertBox.okayMc._visible = false;
        };//End function

        //Decides what to do with the mailResult
        MailForm.onMailResult = function(mailData){
            this.formMc.alertCover.alertBox.phpMsg.htmlText =
String.text2Html(mailData.status, 10, "#333333", 2, "left");
            this.formMc.alertCover.alertBox.okayMc._visible = true;
        };//End function

        MailForm.closeAlert = function(){
            //local var to decide if we can clear the fields
            var clearField;
            //Determine if mail was successfully sent
            //i.e PHP sent a variable called sent = 0 or 1
            //Convert to number because external data is always a string
            if (Number(this.mailData.sent) == 1) clearField = true;

            //Return textField back to editable state
            var i = this.elements.length;
            while (i--) {
                switch (this.elements[i].type) {
                    case "input":
                    case "textArea":
                        this.elements[i].field.selectable = true; //allow editing
                        this.elements[i].field.type = "input"; //allow editing
                        //Should we clear the field?
                        if (clearField) this.elements[i].field.text = "";
                        break;
                }
            }
            //End while loop
            //if mail was sent successfully then focus the first input element
            //all over again in the form..
            if (clearField) this.focusFirstField();

            //delete the old loadVars object
            delete this.mailData;

            this.talking2Server = false;
            this.formMc.alertCover.removeMovieClip();
        };//End function

        //Make the form
        MailForm.init();

```

We now have a Flash email form ready to talk to the server. Let's write a script which can talk back to Flash after doing what it has to do, which in our case is send out email, but of course its never that simple. When dealing with arbitrary input from end-users we have to spend some extra energy doing the obvious Error Checking. Error Checking Our Flash Application is sending out 3 variables, name, email and message. We want the name to be optional, email and message to be required. Hence we will write code to check if these variable are available to us. If not we will throw an error. We will also throw an error if the email address is invalid or it is not in a format that is usable. That error will be in the form of two variable values pairs printed as plain text:

1. status = The error message in plain English
2. sent = 0 to signify that we did not send any email

Sending Email PHP can send email via its mail function and the sending is pretty simple stuff. We will send 2 copies of the message. 1 to the end-user to confirm that we have received their email and the second to ourselves. Now, of course we have to deal with formatting this email and adding few additional notes about it. We use plain text templates with inbuilt custom variables such as {message} (within curly braces) to insert user-data and leave the rest of the language intact. We will read this text file and create a new message out of it. So here goes the script. Look for the **print** and **die** methods and see how this script talks back to Flash in plain text.

```
<?php
//Initialize PHP

//Checks if an email pattern is okay
function emailOK($str) {
    //Check empty
    if(empty($str)) return false;

    //Check for @
    if(!ereg("@",$str)) return false;

    //Check for at least 1 dot
    if(!ereg("\.", $str)) return false;

    //Get a user and a host
    list($user, $host) = explode("@", $str);

    //Make sure we have a user and host
    if((empty($user)) || (empty($host))) return false;

    //These characters are not allowed in email addresses
    $badChars = "[ ]+| |+|=|[]|{}|\(|\)|,|:|!|<|>|%|*|/|'|\\"|~|\?|#|\$|\&|\^|www[.]";
    return !ereg($badChars, $str);
}

//End Function

//This function reads any file and spits out its contents in the return
function readTextFile($file){
    if(!($fp=@fopen($file,"r"))) return false;

    //Read the file
    $fileContent="";
    while (!feof($fp)) {
        $fileContent.= fgets($fp, 1024);
    }
}

//End while loop
```

```

//Close the file
if(!fclose($fp)) return false;

return($fileContent);
} //End function

//Plain text email sending function
function sendMsg($to, $toEmail, $sub, $msg, $from, $fromEmail) {
    //Compose headers for plain text email
    $headers = "MIME-Version: 1.0\r\n";
    $headers .= "Content-type: text/plain; charset=iso-8859-1\r\n";
    $headers .= "X-Mailer: PHP/" . phpversion() . "\r\n"; //The mailer name
    $headers .= "From: ".$from."<".$fromEmail.">\r\n";
    $headers .= "Reply-to: ".$from."<".$fromEmail.">\r\n";

    //Compose recipient
    $recipient = empty($to) ? $toEmail : $to."<".$toEmail.">";
    return mail($recipient, $sub, $msg, $headers);
} //End function

//Cleans up a string by trimming it
//if magic_quotes are on.. then stripslashes as well
function cleanUpData($data){
    $data=trim($data);
    if(get_magic_quotes_gpc()){
        $data=stripslashes($data);
    } //End if
    return($data);
} //End function

//Sends email from Flash Data
function flashEmail($postVars){

    //These variables in $postVars came from Flash
    //They are urlencoded for safe transfers
    //So we have urldecode them and then we trim them
    $name = cleanUpData(urldecode($postVars["name"]));
    $email = cleanUpData(urldecode($postVars["email"]));
    $message = cleanUpData(urldecode($postVars["message"]));

    /*ERROR CHECKING*/

    //Email is required
    if (empty($email)) {
        $status = "Email is empty!\nCannot send message";
        die("status=".urlencode($status)."&sent=0&");
    } //End if

    //Check if email is okay..
    if (!emailOK($email)) {
        $status = "Email is invalid!\nCannot send message";
        die("status=".urlencode($status)."&sent=0&");
    } //End if

    //Something in the message is required
    if (empty($message)) {
        $status = "Message is empty!\nCannot send an empty message";
        die("status=".urlencode($status)."&sent=0&");
    } //end if

```

```

/*#####*/
//Everything looks good.. we are still alive!
//Deal with flash carriage return and convert to new lines
$message = str_replace("\r", "\n", $message);

//The subject line
$subject = "Productbeta Email Tutorial";

//Change this to your information
$ourName = "Productbeta";
$ourEmail = "navneet@productbeta.com";

//Open the template files
$toSenderMsg = readTextFile("senderThanks.txt");
//Replace in-built variables in the senderThanks.txt file
$toSenderMsg = str_replace("{greetings}", empty($name) ? "Hello" : "Dear ".$name.",", $toSenderMsg);
$toSenderMsg = str_replace("{date}", date("m/d/Y"), $toSenderMsg);
$toSenderMsg = str_replace("{message}", $message, $toSenderMsg);

$ourVersion = readTextFile("websiteMail.txt");
$ourVersion = str_replace("{greetings}", "Hi,", $ourVersion);
$ourVersion = str_replace("{date}", date("m/d/Y"), $ourVersion);
$ourVersion = str_replace("{name}", empty($name) ? "none" : $name, $ourVersion);
$ourVersion = str_replace("{email}", $email, $ourVersion);
$ourVersion = str_replace("{message}", $message, $ourVersion);

//Done with all formatting.. let's send our messages
$tTasks = 2;
$tDone = 0;

//sendMsg function arguments?
//sendMsg($to, $toEmail, $subject, $message, $from, $fromEmail)
//Send a copy the website user
if (sendMsg($name, $email, $subject, $toSenderMsg, $ourName, $ourEmail)) $tDone++;

//Send a copy to ourselves
if (sendMsg($ourName, $ourEmail, $subject, $ourVersion, empty($name) ? $email : $name, $email)) $tDone++;

if ($tDone == $tTasks) {
    $status = "Message Sent!\nThank you";
    print "status=".urlencode($status)."&sent=1&";
    return true;
} else {
    $status = "Could not send message because of an internal server error\nPlease try again";
    die("status=".urlencode($status)."&sent=0&");
} //End if
} //End function

//Send Flash Email
flashEmail($_HTTP_POST_VARS);

//Goodbye PHP
?>

```

And with that we have a pretty simple yet powerful email sending application in Flash.