Web Impressions

Because first impressions last

Contents

Introduction

Whilst many of the code examples provided in this guide are applicable to any Windows based web server, the information provided is designed for customers using Fasthosts web servers; some parameters may not be valid on other hosting platforms.

ℹ **Important:** You are responsible for the misuse of any scripts on your website.

Unfortunately we are unable to provide email and telephone support for scripting, but you'll find a wealth of information about Perl at: http://www.perl.com/pub/a/2000/10/begperl1.html

Scripting Perl with JMail

Before we go into detail on how to get your form working, let's look at some ground rules that will ensure your mail isn't blocked:

- Fasthosts redirects all web based mail to an SMTP Filter System.
  Emails must have either a valid "from" or "to" address which is a mailbox hosted with Fasthosts. Any email that does not fulfil this criterion will not be delivered. If you are sending email to a customer who has given you his email address, you need to use the domain name of your website.

- Fasthosts' SMTP Filter System rate limits outgoing mail from any domain, to prevent bulk emailing. Our limits are set to allow normal form-based email activity to pass unhindered, but stop any persistent attempt to send bulk mail.

- The SMTP Filter System will prevent mass emailing (spam).
  All attempts to send bulk emails are logged and may result in the unfortunate suspension of your website, in accordance with our misuse policies.

Perl users should use the JMail object in place of BLAT for sending mail from scripts.

Using JMail in Perl to send email

The first thing to do is to make sure your form in the submitting page is correct. In its simplest form it should look like this:

```
<form action="cgi-bin/jmail.pl" method="post"
name="mailform">
        <input name="email" type="text" size="40">
        <input name="email_submit" type="submit"
value="send mail">
</form>
```

Now to the real scripting (using the example form above, we’re now talking about the code in the “jmail.pl” file).
Note that this script is placed in the “cgi-bin” directory because it has the necessary permissions to execute Perl scripts.

First, initialize the necessary units and set up the JMail object:

```
use OLE;
use CGI;
$jmail = CreateObject OLE "JMail.SMTPMail";
```

Next, output the HTML headers:

```
$form = new CGI;
        $Recipient=$form->param('email');

        $domain = $ENV {'SERVER_NAME'};
        $referer = $ENV {'HTTP_REFERER'};
        $url = $referer;
        $url =~ s/^http:\/\///i;
        $url =~ s/^www\.//i;
        $domain =~ s/^www\.//i;
```

Next, set up the mail object:

```
$Sender = "noreply\@$domain";
        $SMTPServer = "smtp.$domain:25";
        $Subject = "JMail Example";
        $Body = "This test mail sent from:
$ENV{'LOCAL_ADDR'} using the JMail component on the
server via Perl.";
        $Priority=3;
        $Header = "Originating-IP",
$ENV{'REMOTE_ADDR'};

        $jmail->{ServerAddress} = $SMTPServer;
        $jmail->{Sender} = $Sender;
        $jmail->{Subject} = $Subject;
        $jmail->AddRecipient ($Recipient);
        $jmail->{Body} = $Body;
        $jmail->{Priority} = $Priority;
        $jmail->AddHeader ($Header);
```

Next, use this information to check that the posting page is on the same website as the script. This prevents other people from using your script for bulk emailing.

```
if ($url =~ m/^$domain/)
        {
                $mailmessage = "mail sent";
                $jmail->Execute;
        }
        else
        {
                $mailmessage = "mail was not sent.
Incorrect Referer";
        }
```

Finally, output the results:

```
print "Result: $mailmessage  Recipient: $Recipient";
        print "Sender: $Sender  SMTP Server:
$SMTPServer";
        print "Subject: $Subject  Referer: $referer";
        print "Domain: $domain  url: $url ";
```

Important: If you don't protect your scripts by checking the referrer, they will be open to abuse by bulk mailers. If this happens your website will be at risk of being suspended in accordance to our misuse policies.

With Perl, all resources used are destroyed at the end of the script, so you don't need to close and destroy objects you’ve used.

Scripting Perl with an Access database

Whilst the following code examples are applicable to any Windows based web server, the information provided is designed for customers using Fasthosts web servers; some parameters, such as the methods used to derive file paths, may not be valid on other hosting platforms.

Our Perl use the same OLE objects as ASP.

The current version of ADO installed on our servers is MDAC 2.8. Detailed component information can be obtained from Microsoft or one of the many scripting sites on the web.
Tip: Search on Google for "recordset.movenext" or similar.

Before looking at the code, the following points should be considered when using an Access database in a shared hosting environment:

- All scripts run on a shared server, and thus what happens on one website has an effect on the stability of certain components on other websites on the same server. Access ADO is such a component. It is a shared resource, and should therefore be programmed correctly.
- The Jet access engine has a limited set of resources. Once these are exhausted, the script will generate errors. These are normally temporary in nature and can occur when server loading is very heavy. These are operating system limitations, and cannot be extended or isolated by us.
- You should only use Access 2000 or higher databases.
  If you use Access 97 you may receive the following error:-
  [Microsoft][ODBC Microsoft Access Driver] Cannot open database '(unknown)'. It may not be a database that your application recognizes, or the file may be corrupt.

This is due to an issue in MDAC 2.5 and greater, and is only resolved by using a newer version of Access.

- Use as few connections as possible. If you are using session and application based objects, use a single application based connection. Otherwise, use one per script, explicitly opening the connection at the start of the script, and then closing it at the end of the script.

Using ADO in Perl to connect to an Access Database

Now to the code. First initialise the units and output the HTML header:

```
use Win32::OLE;
use Win32::OLE::Const ;

print "Content-type: text/html\n\n";
```

Next, get the web root folder so that we can get the path to the database in the PRIVATE folder.

Next set up the database connection. Always explicitly open the connection. Do not use a connection string with objects such as a command or a recordset object. These implicitly open a database connection which they then leave open to time out after the script closes. This connection accesses the Northwinds database which is stored in the PRIVATE directory below the website root folder.

```
$webroot = substr ($ENV{'PATH_TRANSLATED'}, 0, length
($ENV{'PATH_TRANSLATED'}) - (length ($ENV{'PATH_INFO'})+
(length("htdocs")))).”private";
        print $webroot;

        $DBFile = "$webroot\\northwind.mdb";
        $connstr = "PROVIDER=MSDASQL;DRIVER={Microsoft
Access Driver (*.mdb)};DBQ=$DBFile;UID=;PWD=;";
        print $connstr;

        # Create our ADO Connection object and open it
        $cnn = Win32::OLE->new("ADODB.Connection");
        $rec = Win32::OLE->new("ADODB.Recordset");
        $cnn->Open($connstr);
```

Now open the resordset using the current connection. In this case we are querying the customer table.

```
$SQL ="SELECT top 10 (city) FROM customer";

        $rec->open($SQL, $cnn, 1, 1);
```

With the recordset now open, we iterate through it and output the results to the page. Note that we don't need to use the movefirst method as the recordset points to the first record by default when opened. With an empty recordset, the loop exits immediately without errors.

```
until($rec->EOF)
        {
          $fields = $rec->Fields;
          $firstname=$fields->Item("city")->value;
          print $firstname;
          $rec->MoveNext();
        }
```

Finally, we close the connection.

```
$cnn->Close;
```

Further reading

Unfortunately we are unable to provide email and telephone support for scripting, but there is a range of information available online to help you along the way.

External links to perl.com

- Beginner’s Introduction to Perl
- Database programming with Perl
- Ten essential development practices
- How to avoid writing code