

FormMail Application for ASP.Net

Author:
Alex Gust

Course:
Web Systems, CPET 499

Professor:
Paul Lin

Executive Summary

Background

A form mail application can be used alongside static web sites or sites with client-side logic only to allow forms to be processed and sent via email to a specified recipient.

Forms are typically processed using server-side logic in the form of PHP, ASP.Net, or similar technologies. Many simple web sites, however, also need the ability to collect information via web forms. A form mail application is a self-contained script or program that resides in an executable directory of the web server, and can accept POST data from an HTML page. Once the POST data is received, the form mail application will compile the information in a readable format and then email it to a specified recipient.

Existing Technologies

There are many form mail applications available on the web, including ones for ASP, but none were found based on ASP.Net, specifically version 4. One was found based on VB Script, however Microsoft has discontinued supporting VB Script, their most recent server-side scripting language.

Most popular existing FormMail scripts on the web, and reasons for obsolescence:

- Matt's FormMail
Uses Perl, No longer developed, Insecure
<http://www.scriptarchive.com/formmail.html>
- NMS FormMail
Uses Perl, last updated in 2006
<http://nms-cgi.sourceforge.net/>
- ASP FormMail
Uses classic ASP and VBScript, technologies no longer supported by Microsoft
<http://www.brainjar.com/asp/formmail/default3.asp>

Purpose

The purpose of this project is to build a portable, reusable form mail application based on C# and the windows hosting platform running .Net 4. Newer .Net libraries are to be utilized (in the System.Net.Mail namespace). The completed FormMail application shall be published to the Internet and made available for free download.

Scope

1. The FormMail Application is built as a .Net application consisting of a single uncompiled ASPX file containing all code.
2. The application is user configurable based on either POST data, or configuration variables contained at the top of the ASPX file.
 - a. User may specify a set of email recipients to which the form data shall be emailed.
 - b. User must specify an SMTP server including username, password and port number.
 - c. User may specify a return address, subject line for the email.
 - d. User may specify form fields that are required for form to be submitted.

- i. Script shall display built in error messages by default.
 - e. User may specify HTML pages to which the user is redirected after successful or failed form submission.
- 3. The application shall support external SMTP servers with authentication, or via the local server (assuming it too supports SMTP with authentication).
- 4. Installation procedure is simple – a single FormMail.aspx file is copied to the destination web server.
 - a. FormMail script works with only a minimum of 5 user configured options.
- 5. Security measures must be implemented to prevent the FormMail application from being abused by spammers or anyone external to the web server on which it runs.
- 6. The form mail application shall have a limited user-facing component. It will display error messages and will support a custom error page in the event of missing fields by means of redirection to a user specified page.
- 7. The application shall be accommodating to Text Boxes, Radio Buttons, Drop Down Lists and Check Boxes on the user form. Small to medium sized forms are supported (Approximately 5 – 20 fields).
- 8. Emailed output shall be in HTML format and viewable on both desktop and mobile email clients. Mobile design will implement a reflexive CSS design to display without horizontal scrolling.

DELIVERABLES

- 1. Working prototype
 - a. Uncompiled ASPX file
 - b. Commented Source Code
- 2. Installation Instructions
- 3. Integration Instructions
- 4. Sample HTML Form
- 5. Blog post (thegustfactor.blogspot.com)
 - a. Project description
 - b. Installation Instructions
 - c. Integration Instructions
 - d. Sample HTML Forms
 - 1) Including both a basic, “quick start” version and an advanced implementation showing all options.
 - e. Downloadable ASPX file

1.Web System Design

Architecture

This web system encompasses two out of the three tiers of a Three Tier Architecture design. The third tier is considered as belonging to the end-user email recipient.

- Client Tier: Consists of the display elements of the FormMail ASPX page, including all user-facing components of the FormMail.aspx page
 - Error Messages directed at the end-user
 - Error Messages relating to improper configuration, directed at the web developer
 - Emails sent to the form owner
- Business Logic components of the FormMail.aspx page
 - Routine to read POST data from the HTTP stream
 - Routine to validate POST data (ensuring no malicious code)
 - Routine to validate user configured variables
 - HTML Generation routine, to build the email message
 - Email sending routine
 - Handles SMTP authentication
 - Mail delivery
 - Error checking
- Database Tier discussion
 - The database tier is considered the form owner's email box. Data is not written to the developer's web server in any way, and the results of the form mail will be stored as email messages.

Software Tools Selection

Tools were selected based on compatibility with the intended platform for this project. All tools were in the Microsoft suite of web development software.

- a. Visual Studio 2012
 - i. Purpose: IDE For Code Development
- b. Windows Server 2012
 - i. Purpose: Operating System that can run IIS
- c. IIS 8
 - i. Purpose: Compile and display ASPX files

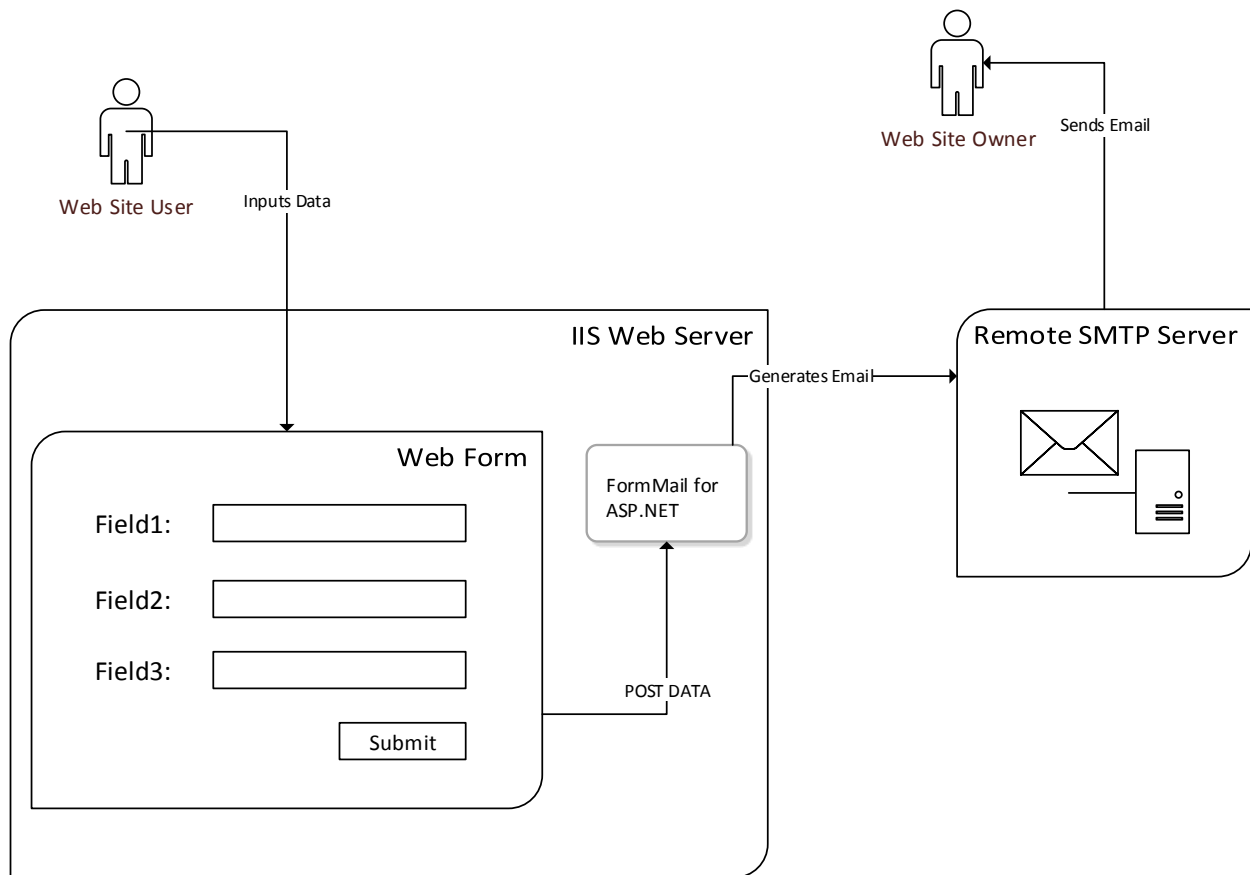
System Hardware Specification

The FormMail ASPX file can run on any Windows web server running IIS and .Net extensions version 3.5 or higher. Additionally, an externally hosted SMTP provider was chosen (Gmail), to test email delivery.

- a. Web accessible with a static IP to host a sample form as well as the form mail application.
- b. Windows Virtual Machine
 - ii. 2GB Ram, Xeon 2.4ghz processor
 1. Needed only for development
 - iii. About 50MB of Web Bandwidth required

iv. Disk storage 20GB (Operating System and Web Site Files)

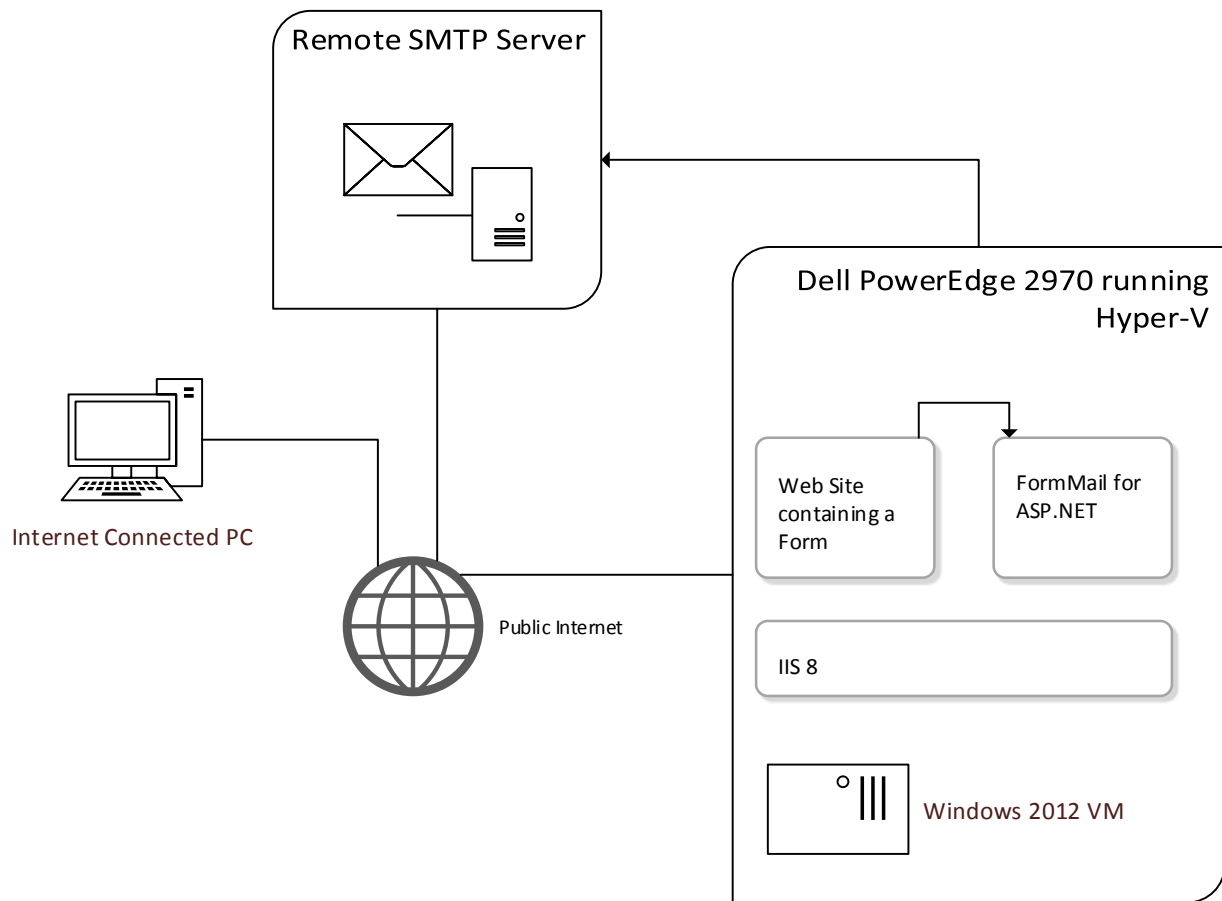
UML Use Case / System Interaction Diagram



FormMail Operation Description:

A web developer will install the FormMail ASPX file and link it to a web form. The end user of the web site will access the web form, provide data, and click Submit. The FormMail ASPX application will parse the data and send it to a remote SMTP server, which will then deliver an email to the web site owner.

Web System State Diagram



Description: The test environment will emulate a potential environment in which the FormMail ASPX file may be installed. In this case, it is a web page, hosted by IIS8, running under a Windows 2012 virtual machine. The machine is Internet Connected, and linked with a Remote SMTP Server. Any Internet connected PC may access the test site and use it to generate email messages.

Web System Integration

Hardware Configuration and Validation

1) Development Machine

- a. Windows Server 2008R2 running Visual Studio 2013 was used. System specifications

Windows edition	
Windows Server 2008 R2 Standard	
Copyright © 2009 Microsoft Corporation. All rights reserved.	
Service Pack 1	
System	
Processor:	Intel(R) Xeon(R) CPU E31245 @ 3.30GHz 3.29 GHz
Installed memory (RAM):	6.70 GB
System type:	64-bit Operating System
Pen and Touch:	No Pen or Touch Input is available for this Display



2) Testing Machine

- a. Windows Server 2012R2 running IIS8 was used.

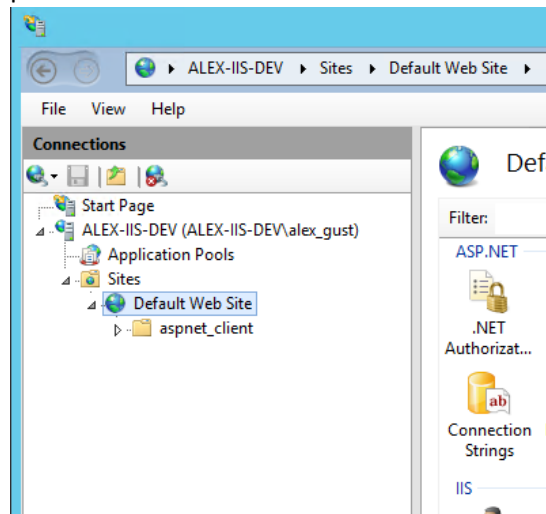
Windows edition	
Windows Server 2012 R2 Standard	
© 2013 Microsoft Corporation. All rights reserved.	
System	
Processor:	Quad-Core AMD Opteron(tm) Processor 2372 HE 2.10 GHz
Installed memory (RAM):	3.91 GB
System type:	64-bit Operating System, x64-based processor
Pen and Touch:	No Pen or Touch Input is available for this Display



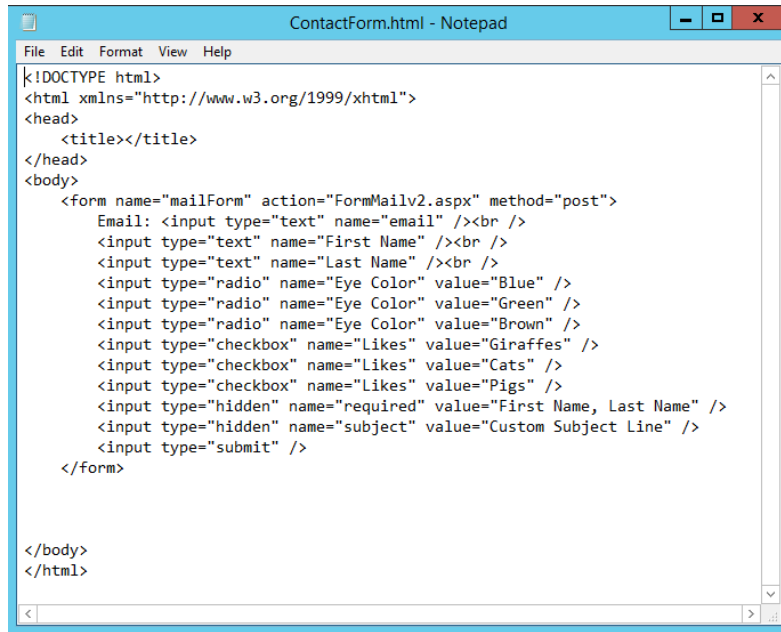
Web System / Server Configuration

1) Testing Machine

- a. IIS8 Was installed with default options. Default Web Site was created with bindings on port 80



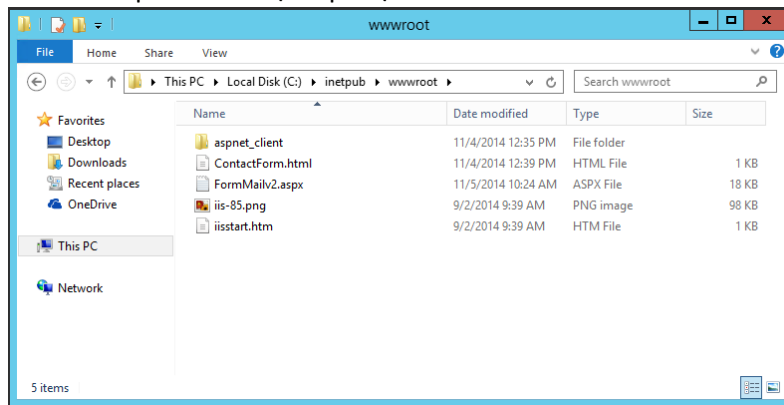
- b. Sample HTML Form was created:



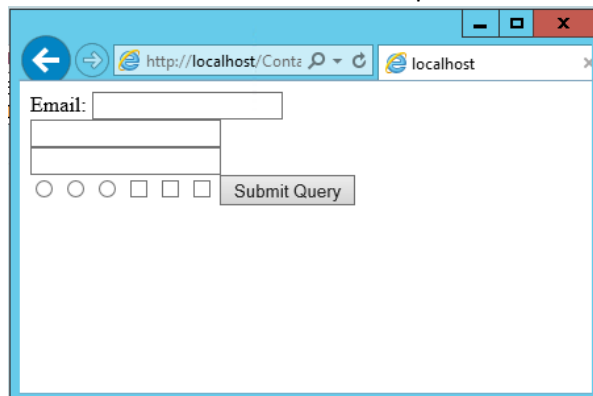
```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title></title>
</head>
<body>
<form name="mailForm" action="FormMailv2.aspx" method="post">
  Email: <input type="text" name="email" /><br />
  <input type="text" name="First Name" /><br />
  <input type="text" name="Last Name" /><br />
  <input type="radio" name="Eye Color" value="Blue" />
  <input type="radio" name="Eye Color" value="Green" />
  <input type="radio" name="Eye Color" value="Brown" />
  <input type="checkbox" name="Likes" value="Giraffes" />
  <input type="checkbox" name="Likes" value="Cats" />
  <input type="checkbox" name="Likes" value="Pigs" />
  <input type="hidden" name="required" value="First Name, Last Name" />
  <input type="hidden" name="subject" value="Custom Subject Line" />
  <input type="submit" />
</form>

</body>
</html>
```

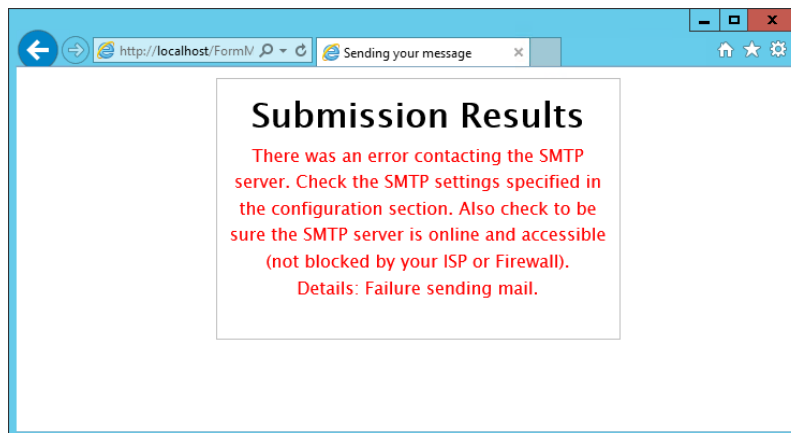
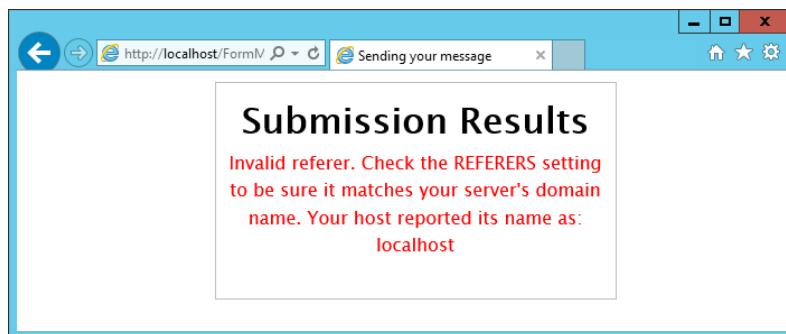
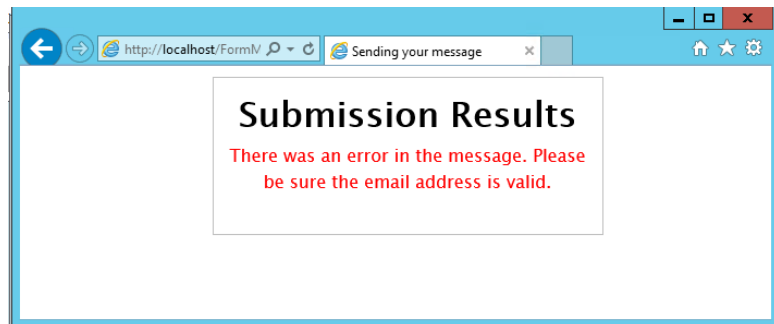
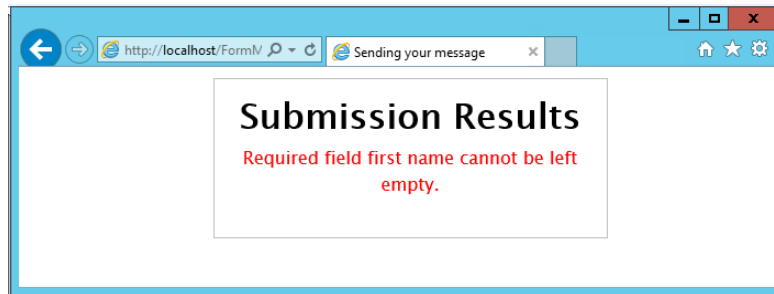
c. Files were placed at c:\inetpub\wwwroot



d. Form was accessed via Internet Explorer web browser



e. Error Screens were tested



CSS Design

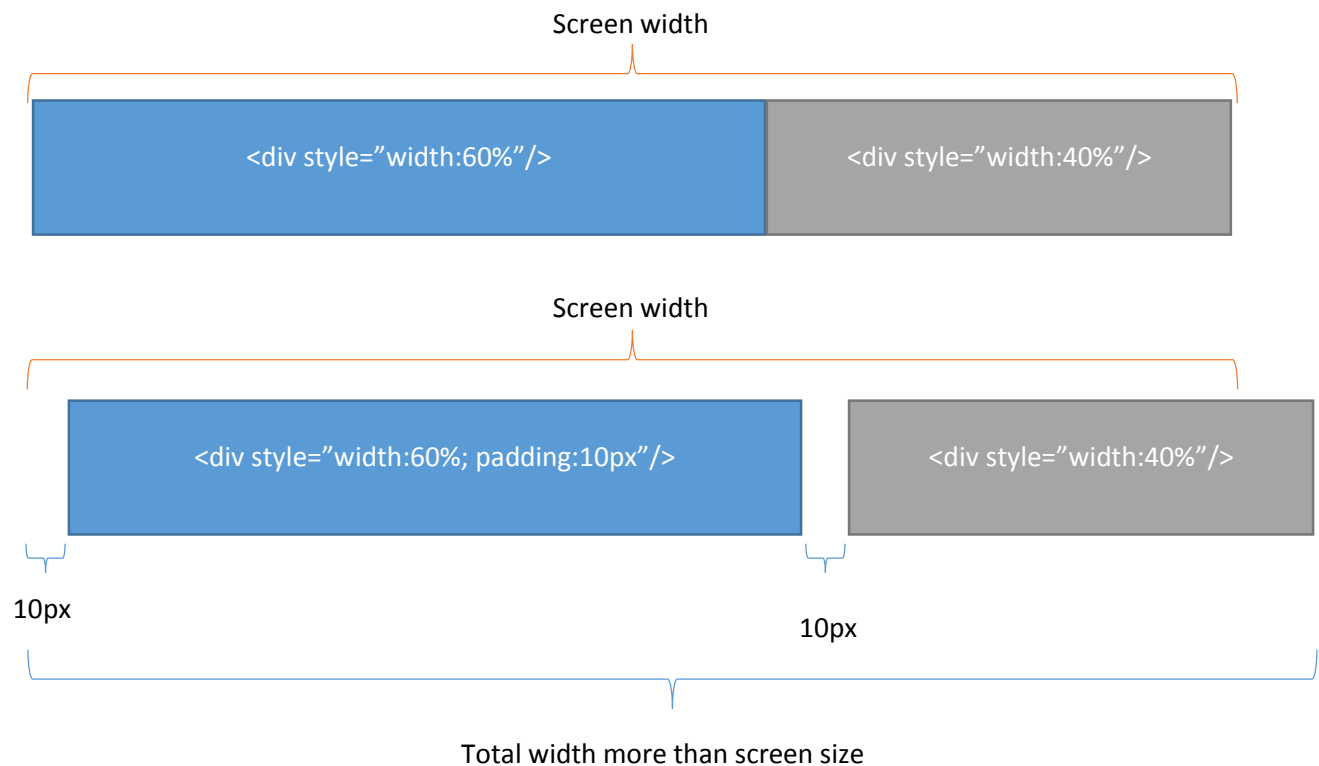
At first thought, one would think to accomplish this by adding two DIV's, one on the left, and one on the right, applying padding and margins to keep the text away from other elements.

However, due to the need for a reflexive design, widths must be specified in percent rather than fixed amounts, so they will resize with the browser window.

As always, if two blocks exceed the width of the screen, they will wrap around to the next line

Padding and Margin properties were found to cause elements to wrap when added along with percentage based width properties.

For example, two elements, at 40% width and 60% width will fill the width of the screen while it is resized in any fashion. However, a 40% element, and a 60% width element with 10px of padding (or margin) will wrap to the next line. Taking an example screen, 800 pixels wide, the 40% element will occupy 320 pixels, and the 60% element will occupy 480 pixels, plus 10px margins on both the left and right sides, for a total of $320\text{px} + 480\text{px} + 20\text{px} = 820\text{px}$, which guarantees the second div wraps to the next line.



HTML Design

Two sample forms were created, the first demonstrating the simplest way to use the FormMail script, and the second demonstrating all available options.

Simple Form:

```
<form name="mailForm" action="FormMail.aspx" method="post">

    First Name: <input type="text" name="First Name" /><br />
    Last Name: <input type="text" name="Last Name" /><br />
    <input type="submit" />

</form>
```

All available options:

```
<form name="mailForm" action="FormMail.aspx" method="post">

    First Name: <input type="text" name="First Name" /><br />
    Last Name: <input type="text" name="Last Name" /><br />
    Email: <input type="text" name="Email" /><br />

    <br />

    Eye Color: <br />
    <input type="radio" name="Eye Color" value="Blue" />Blue
    <input type="radio" name="Eye Color" value="Green" />Green
    <input type="radio" name="Eye Color" value="Brown" />Brown
    <br /><br />

    Animals you like:<br />
    <input type="checkbox" name="Likes" value="Giraffes" />Giraffes
    <input type="checkbox" name="Likes" value="Cats" />Cats
    <input type="checkbox" name="Likes" value="Pigs" />Pigs
    <br /><br />

    Favorite car:<br />
    <select name="Favorite Car">
        <option value="">None</option>
        <option value="volvo">Volvo</option>
        <option value="saab">Saab</option>
        <option value="fiat">Fiat</option>
        <option value="audi">Audi</option>
    </select>

    <input type="hidden" name="subject" value="Custom Subject Line" />
    <input type="hidden" name="redirect" value="Success.html" />
    <input type="hidden" name="missing_fields_redirect" value="MissingFields.html" />
    <input type="hidden" name="required" value="Last Name, Email, Likes" />

    <input type="submit" />

</form>
```

Documentation was created, explaining all of the options available in the FormMail ASPX file:

```
const String DEFAULT_FROM_ADDRESS = "noreply@mydomain.com";
```

The address that the email will appear to be from. This is a special field that can be overridden by a field named "Email" in your form. If you make this a text field, you can allow the user to provide their email address and then you can use your email client's Reply function to start an email back to them. This field will default to noreply@yourservername if left blank.

```
const String SUBJECT = "A Custom Subject Line";
```

The subject of the email message. Will default to "Form Submission" if left blank.

```
const String REDIRECT = "Success.html";
```

A page on your web server to redirect to after the form is successfully submitted. This can be a URL of any kind. If left blank, the FormMail script will display a success message.

```
const String MISSING_FIELDS_REDIRECT = "MissingFields.html";
```

A page on your web server to redirect to if some required fields were left out (As specified in the REQUIRED string, below). This can be a URL of any kind. If left blank, the FormMail script will display a message indicating which fields were missing.

```
const String REQUIRED = "First Name, Last Name, Email";
```

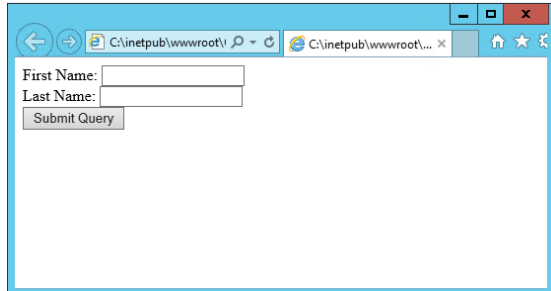
Specify the names of any fields that must be filled in before submitting the form. Comma separate multiple values. These must match the "name=" property of the fields on your web form.

```
const Boolean SHOW_ERRORS = false;
```

Show ASP.Net error messages. This should be used for debugging only, and should be reset to "false" after debugging is complete.

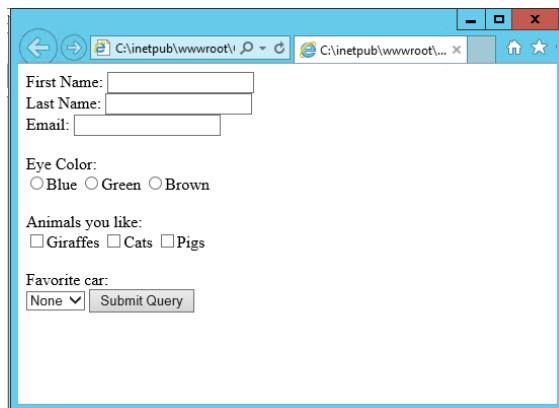
Web System Demonstration and Acceptance Testing

Simple Contact Form:



A screenshot of a web browser window showing a simple contact form. The form has two text input fields labeled "First Name:" and "Last Name:". Below these fields is a "Submit Query" button. The browser's address bar shows the URL "C:\inetpub\wwwroot\...".

Detailed form demonstrating all options:

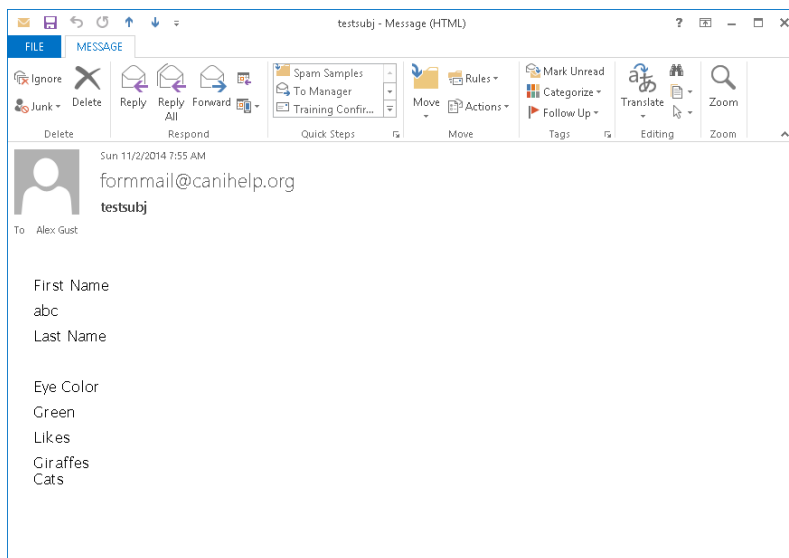


A screenshot of a web browser window showing a detailed contact form. The form includes the following fields and options:

- First Name:
- Last Name:
- Email:
- Eye Color: ☐ Blue ☐ Green ☐ Brown
- Animals you like: ☐ Giraffes ☐ Cats ☐ Pigs
- Favorite car:

The browser's address bar shows the URL "C:\inetpub\wwwroot\...".

Output as displayed on iPhone email client and Output as displayed on Microsoft Outlook Email client:

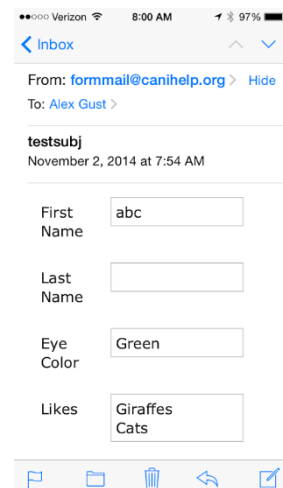


A screenshot of the Microsoft Outlook email client interface. The email is from "formmail@canihelp.org" with the subject "testsubj". The email content is as follows:

Sun 11/2/2014 7:55 AM
formmail@canihelp.org
testsubj
To: Alex Gust

First Name
abc
Last Name

Eye Color
Green
Likes
Giraffes
Cats



A screenshot of an iPhone email client interface. The email is from "formmail@canihelp.org" with the subject "testsubj". The email content is as follows:

From: formmail@canihelp.org
To: Alex Gust

testsubj
November 2, 2014 at 7:54 AM

First Name: abc
Last Name:
Eye Color: Green
Likes: Giraffes, Cats

Requirements Validation

Requirement	Validation Result
The FormMail Application shall be built as a .Net application , either with a compiled DLL file , or, preferably as a single uncompiled ASPX file containing all code.	Project has been delivered as a single, uncompiled ASPX file.
The application shall be user configurable based on either POST data, or a text configuration file.	FormMail is configurable via hidden form fields or setting variables at the top of the code.
The application shall support external SMTP servers with authentication, or via the local server (assuming it too supports SMTP with authentication)	FormMail was tested with gmail.com, hostedemail.com, and canihelp.org SMTP servers, both with and without SMTP authentication, and SSL requirements.
Installation procedure should be simple – i.e. copying files to a web server and setting appropriate execute permissions.	Single file copy required for installation.
Security measures shall be implemented to prevent the FormMail application from being abused by spammers or anyone external to the web server on which it runs.	FormMail checks HTTP referrer property to ensure it is not used cross-site. Also hides recipient email address and SMTP credentials in compiled code. (Not visible from View Source in web browser)
The form mail application shall not have a user-facing component, but shall make its data available in the form of a further POST or GET request, or by means of redirection to a user specified page.	User redirection was implemented. Further POSTing of data was deemed too difficult for novice web programmers to implement, and requirement was replaced with a limited user interface for displaying error messages.
The application shall be accommodating to Text Boxes, Radio Buttons, and Check Boxes on the user form. Small to medium sized forms are supported (Approximately 5 – 20 fields).	All were tested, including forms with up to 20 fields.
Emailed output shall accommodate short text strings, long text strings (Greater than 254 characters, but less than 10,000 characters), and multiple entry controls (Such as Radio Buttons).	Tested good with short and long strings, as well as multiple entry controls. Reads each POST variable and iterates through its, possibly multiple, values.
Emailed output shall be in HTML format and viewable on both desktop and mobile email clients. Mobile design will implement a reflexive CSS design to display without horizontal scrolling.	CSS Design tests good on Microsoft Outlook 2013, Gmail, Yahoo Mail, and iPhone Integrated Mail.

Lessons Learned

CSS for email is much more difficult than designing CSS for standard web browsers. Email clients do not support CSS in a uniform fashion.

Conclusions and Recommendations

FormMail for ASPX was successful, and all necessary requirements were met. Some were changed due to reconsideration of easy deployment for web developers. This project should make creation of web forms much easier for the novice web developer.