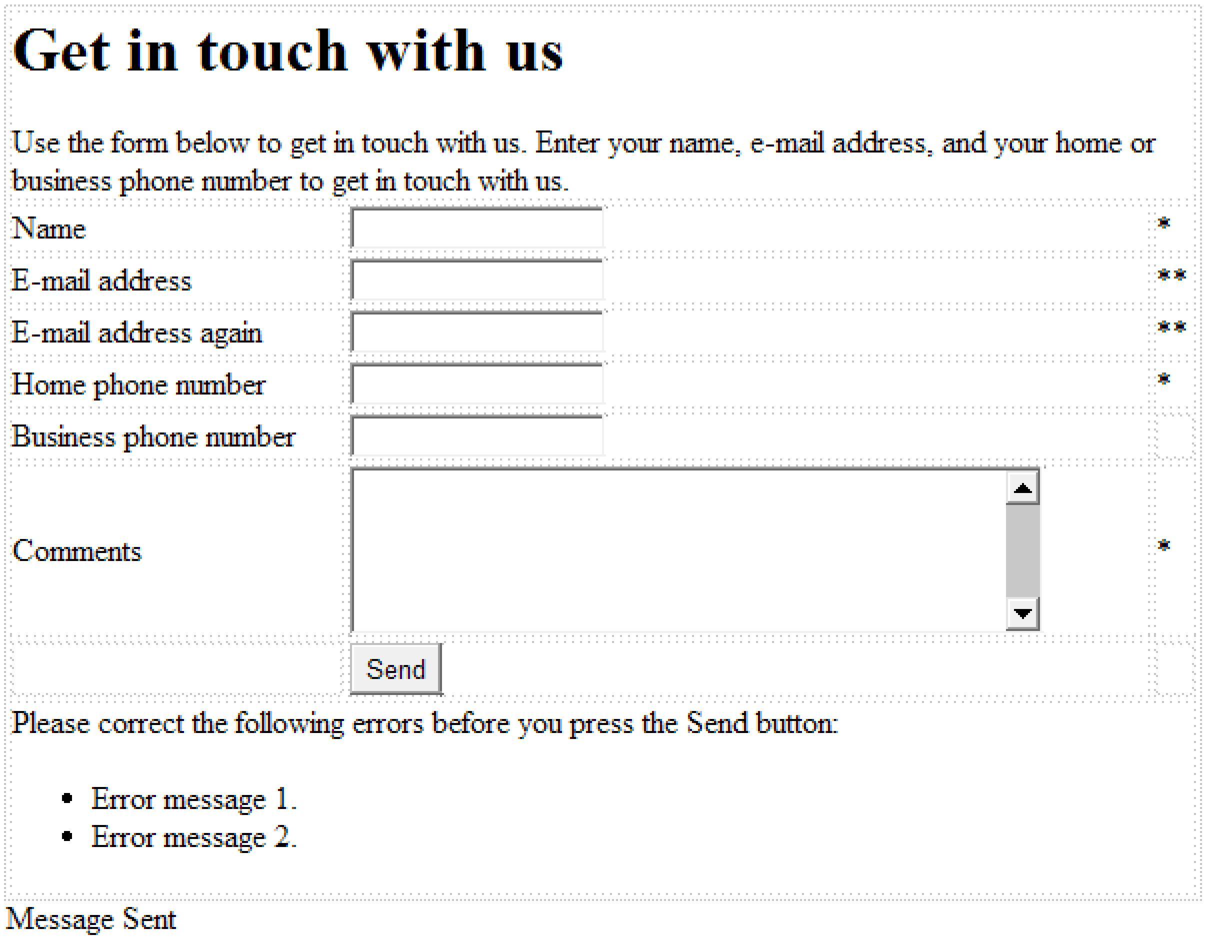
Converting a Contact Form to an eMail Form

You should by now be comfortable with using a form to handle user input. Your task now is to create a form with user information which will be sent via email. The first task obviously will be to create a form on your website which will be used for this function. The form part should look like:



Get in touch with us

Use the form below to get in touch with us. Enter your name, e-mail address, and your home or business phone number to get in touch with us.

Name		*
E-mail address		**
E-mail address again		**
Home phone number		*
Business phone number		
Comments		*
	Send	

Please correct the following errors before you press the Send button:

- Error message 1.
- Error message 2.

Message Sent

This should look very familiar. A table is used to layout the controls which are validated before processing. The code for this form will be provided. It is a rather simple page. Change the page to use a master page for our Halloween Store.

To this point the creation and use of the form is pretty straightforward. What will be unique for this work is the use of the data the user provides to create an email. Before we get to handle the user input we need to build a shell for our data to be placed in, the body of the email.

We will place a file in the App_Data directory that will be used by our email code to form the email from the data provided. The content of ContactForm.txt is:

```
A user has left the following feedback at the site:

Name:              ##Name##
E-mail address:    ##Email##
Home phone:        ##HomePhone##
Business phone:    ##BusinessPhone##
Comments:          ##Comments##
```

The thing that jumps out at you should be the hashed content. This is a way that we will to be able to insert our user data into the content and then send this text as our message body. Let's look at how to make this happen.

Here is the code in total, I'll explain what each section does after.

```
protected void SendButton_Click(object sender, EventArgs e)
  {
    if (Page.IsValid)
    {
      string fileName = Server.MapPath("~/App_Data/ContactForm.txt");
      string mailBody = File.ReadAllText(fileName);

      mailBody = mailBody.Replace("##Name##", Name.Text);
      mailBody = mailBody.Replace("##Email##", EmailAddress.Text);
      mailBody = mailBody.Replace("##HomePhone##", PhoneHome.Text);
      mailBody = mailBody.Replace("##BusinessPhone##", PhoneBusiness.Text);
      mailBody = mailBody.Replace("##Comments##", Comments.Text);

      MailMessage myMessage = new MailMessage();
      myMessage.Subject = "Response from web site";
      myMessage.Body = mailBody;

      myMessage.From = new MailAddress("you@yourprovider.com", "Sender Name");
      myMessage.To.Add(new MailAddress("you@yourprovider.com", "Receiver Name"));

      SmtpClient mySmtpClient = new SmtpClient();
      mySmtpClient.Send(myMessage);

      Message.Visible = true;
      FormTable.Visible = false;
    }
  }
```

After we check that the page is a valid source, we read in the contact response shell. This is done by finding the file and then reading it in to the string mailBody.

Once we have the text read in we need to substitute the hashed entries with real data. That is done by using a string function, Replace.

```
      mailBody = mailBody.Replace("##Name##", Name.Text);
      mailBody = mailBody.Replace("##Email##", EmailAddress.Text);
      mailBody = mailBody.Replace("##HomePhone##", PhoneHome.Text);
      mailBody = mailBody.Replace("##BusinessPhone##", PhoneBusiness.Text);
      mailBody = mailBody.Replace("##Comments##", Comments.Text);
```

Once the data has been injected into the mailBody text we can form our actual email message using ASP.NET mail objects. The following code creates a MailMessage object and then sets the Subject line and the Body of the message. Always include a Subject line in email as many systems will reject email without a subject element.

```
      MailMessage myMessage = new MailMessage();
      myMessage.Subject = "Response from web site";
      myMessage.Body = mailBody;
```

Now we need to make some judgements for the purposes of the assignment. The next section of code uses real email addresses to send our email. To test this process, use your favorite email account, one you check regularly. Be aware however that some email systems might not accept your email because we will be relaying the email (more about that in a moment).

```
      myMessage.From = new MailAddress("you@yourprovider.com", "Sender Name");
      myMessage.To.Add(new MailAddress("you@yourprovider.com", "Receiver Name"));
```

Finally, we need to send the email. To do this we again use an ASP.NET mail object, the SmtpClient. The code is very straightforward.

```
      SmtpClient mySmtpClient = new SmtpClient();
      mySmtpClient.Send(myMessage);
```

With that, our email is on the way—almost. Because we are working with an academic environment we cannot use the school's email system for testing. To actually send the email we need to use a server I have to "relay" the email. This will involve setting the web.config file to contain information about the relay server.

Open your web.config file and go to the bottom after the </system.web> closing tag. Enter the following section after this tag.

```
  <system.net>
      <mailSettings>
         <smtp deliveryMethod="Network" from="student@geeks2b.com" >
            <network host="smtpout.secureserver.net" port="80" userName="student@geeks2b.com"
password="Student2012" />
         </smtp>
      </mailSettings>
   </system.net>
```

Do NOT change the email address in this section as it is used for authentication on the relay server.

With that, your email is on its way!