

# Getting Started with Localizer for ASP.NET

Localizer for ASP.NET from [TMG Development Ltd](#) extends the Microsoft Visual Studio .NET design environment to give design time support for creating localized versions of your ASP.NET WebForms.

## Overview

In this document we will outline the basic steps necessary to produce an internationalized version of your web application using Localizer.

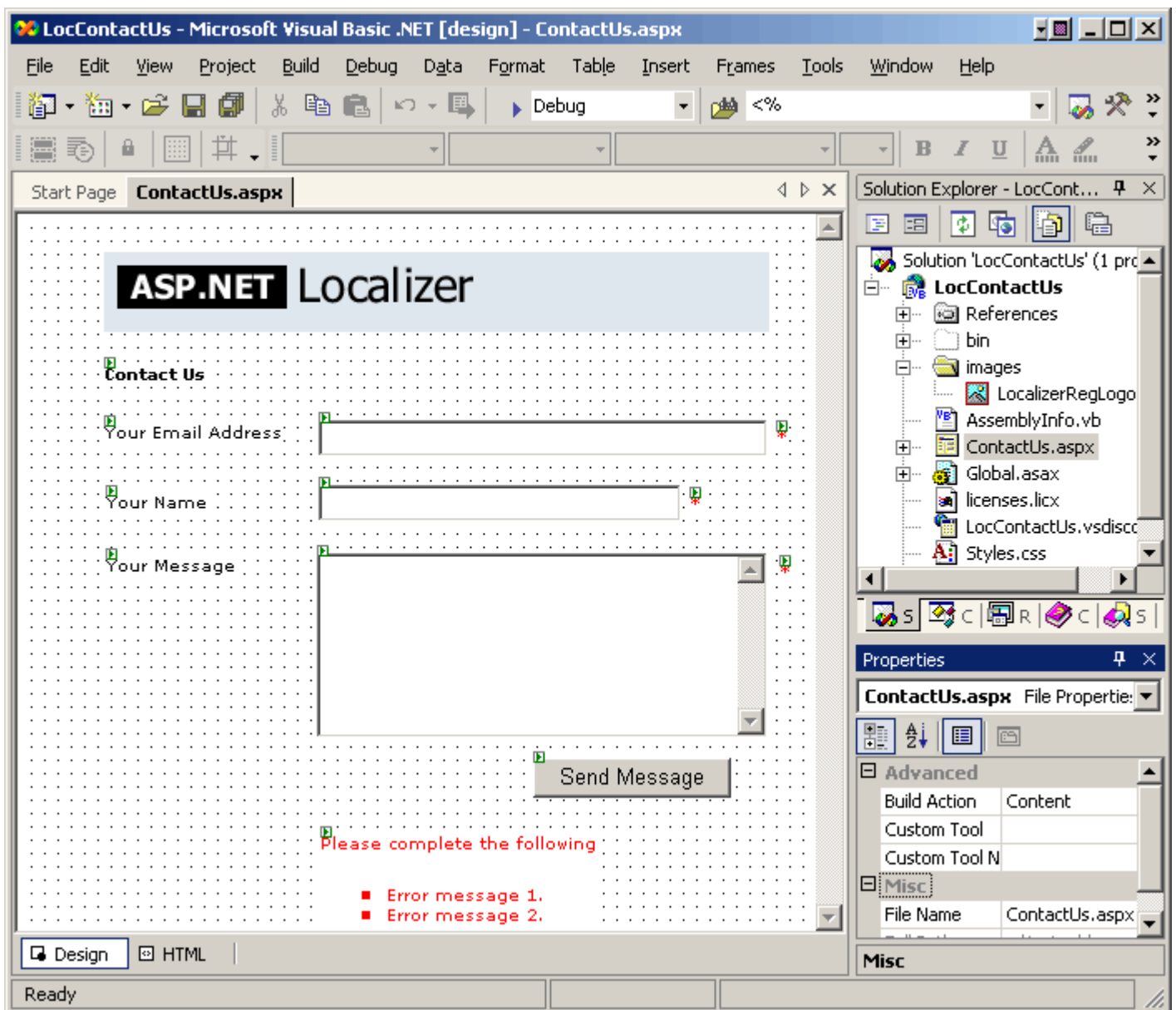
We're going to create a very simple web application that consists of a typical 'Contact Us' page, and then show how we add a Localizer component, customize the text for a couple of languages and set localizer so that it automatically delivers a page in the chosen language.

## Using Localizer at Design Time

To start with, we've created a basic page with a number of WebControls to make up our Contact Us page.

Note that in order for Localizer to be able to localize the properties of a control it must be WebControl based, because at runtime Localizer will need to dynamically set the localized version of the property into the control from server-side code.

So on this form we have some Labels (System.Web.UI.WebControls.Label), TextBoxes (System.Web.UI.WebControls.TextBox) and a submit Button (System.Web.UI.WebControls.Button). We've also added some Validators (System.Web.UI.WebControls.RequiredFieldValidator and System.Web.UI.WebControls.ValidationSummary).



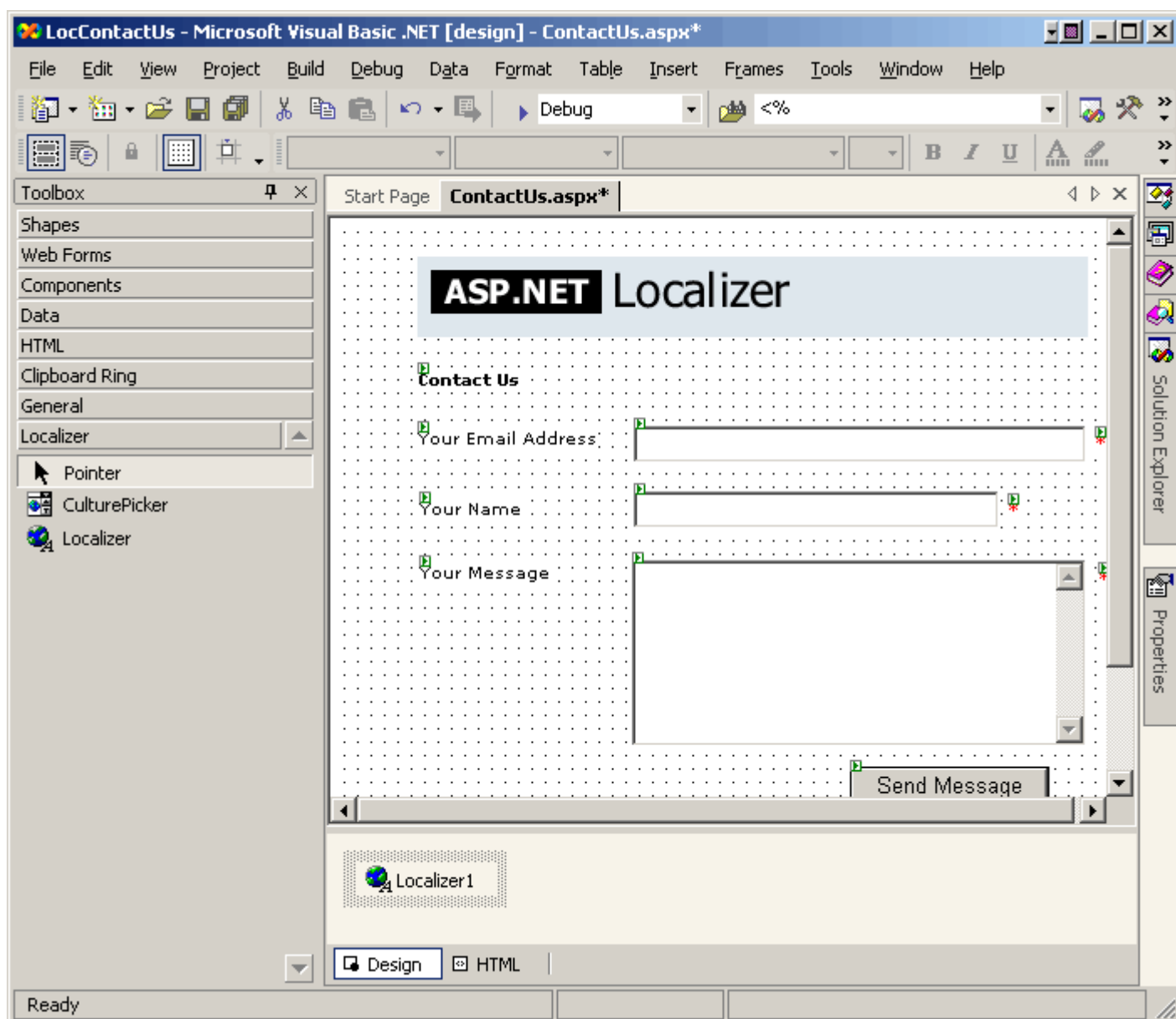
At the moment none of the properties of the web controls on the form are stored in a .resx file - they are all stored in the HTML of the ContactUs.aspx page.


In order to be able to build some culture-specific satellite resource assemblies so that we can load localized resources dynamically at runtime, we need to somehow get all the properties we want to localize out of the aspx and in to the resx. This process will be familiar to Windows Forms programmers because this is exactly what the Windows Form's Localizable property does. As soon as you mark a Windows Form as Localizable by setting Localizable = true, the design environment ensures that all localizable properties are serialized out to a resx file rather than into the Forms designer code section.

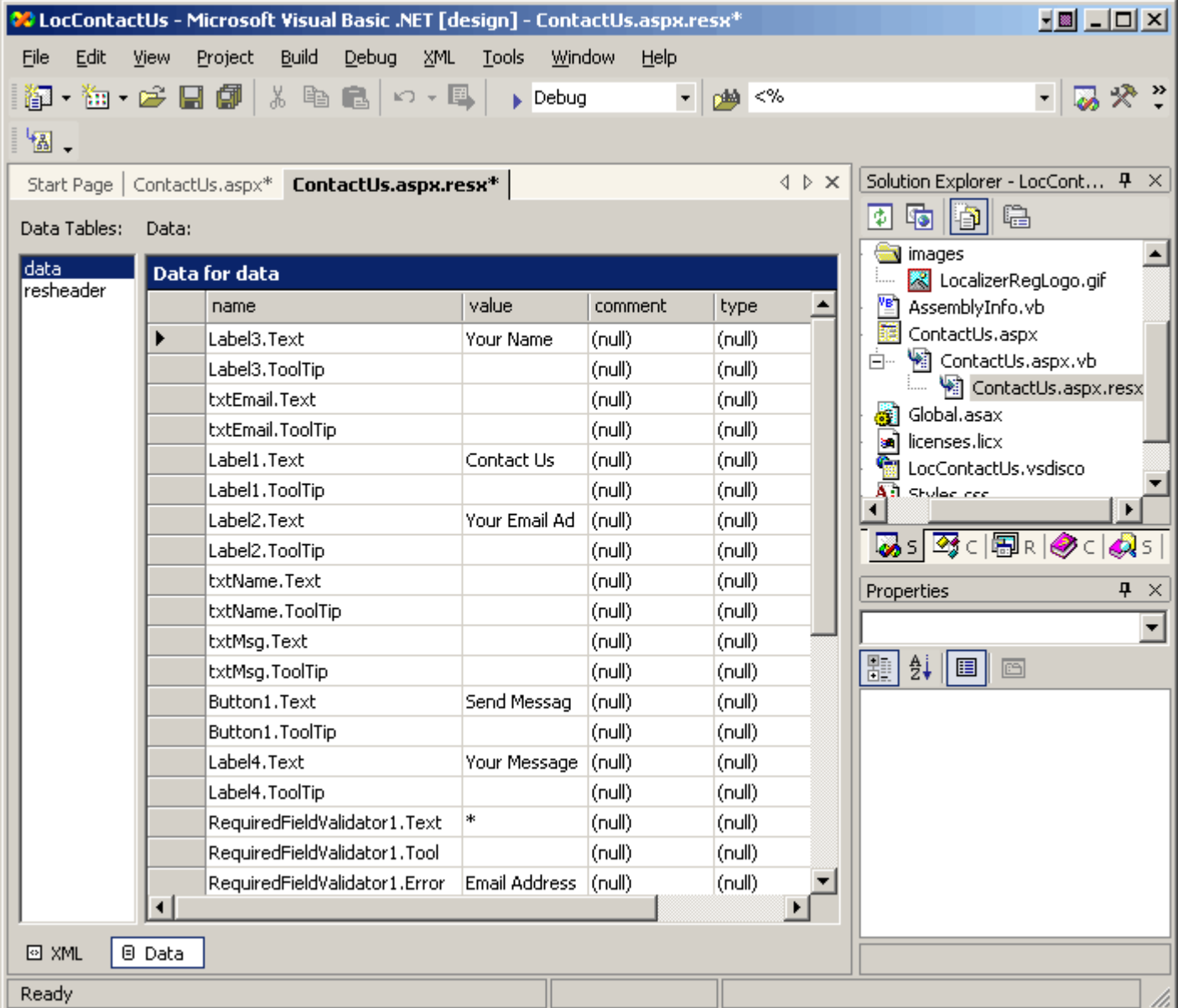
Out of the box, this localization feature is not available for Web Forms. So if you want to produce a localized Web Forms application you have to do all .resx file creation manually. For even a small site this rapidly becomes a daunting task, see the GotDotNet site for details: <http://samples.gotdotnet.com/quickstart/asplus/doc/resourcefiles.aspx>.

This is where Localizer for ASP.NET comes in. It's a standard component that sits in the Visual Studio Component Tray and extends your Web Form to provide the same localization features that Windows Forms programmers have become used to.

So, on with our sample page... we drag a Localizer onto the page and it appears in the component tray:



We can see the resx file entries created by Localizer if we click the 'Show All Files' button  on the Solution Explorer toolbar and drill down into the files under ContactUs.aspx to ContactUs.aspx.resx. Notice when we open this resx file how Localizer has added entries for all the properties on the various controls it recognizes as Localizable:



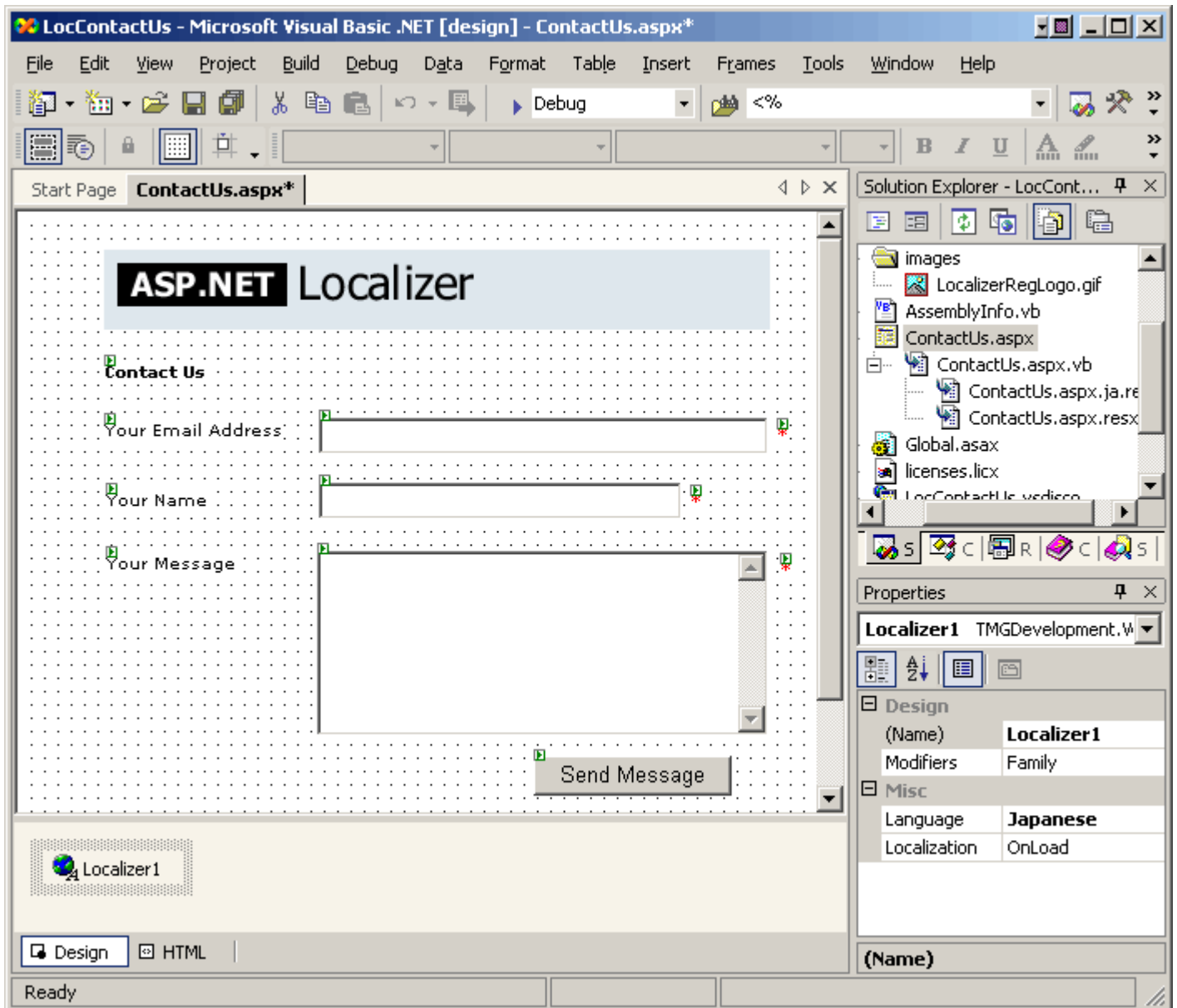
The screenshot shows the Visual Studio IDE with the 'ContactUs.aspx.resx\*' file open. The 'Data Tables' pane displays a table of localized data for the 'data' table. The table includes columns for name, value, comment, and type. The data is as follows:

Data for data				
	name	value	comment	type
▶	Label3.Text	Your Name	(null)	(null)
	Label3.ToolTip		(null)	(null)
	txtEmail.Text		(null)	(null)
	txtEmail.ToolTip		(null)	(null)
	Label1.Text	Contact Us	(null)	(null)
	Label1.ToolTip		(null)	(null)
	Label2.Text	Your Email Ad	(null)	(null)
	Label2.ToolTip		(null)	(null)
	txtName.Text		(null)	(null)
	txtName.ToolTip		(null)	(null)
	txtMsg.Text		(null)	(null)
	txtMsg.ToolTip		(null)	(null)
	Button1.Text	Send Messag	(null)	(null)
	Button1.ToolTip		(null)	(null)
	Label4.Text	Your Message	(null)	(null)
	Label4.ToolTip		(null)	(null)
	RequiredFieldValidator1.Text	*	(null)	(null)
	RequiredFieldValidator1.Tool		(null)	(null)
	RequiredFieldValidator1.Error	Email Address	(null)	(null)

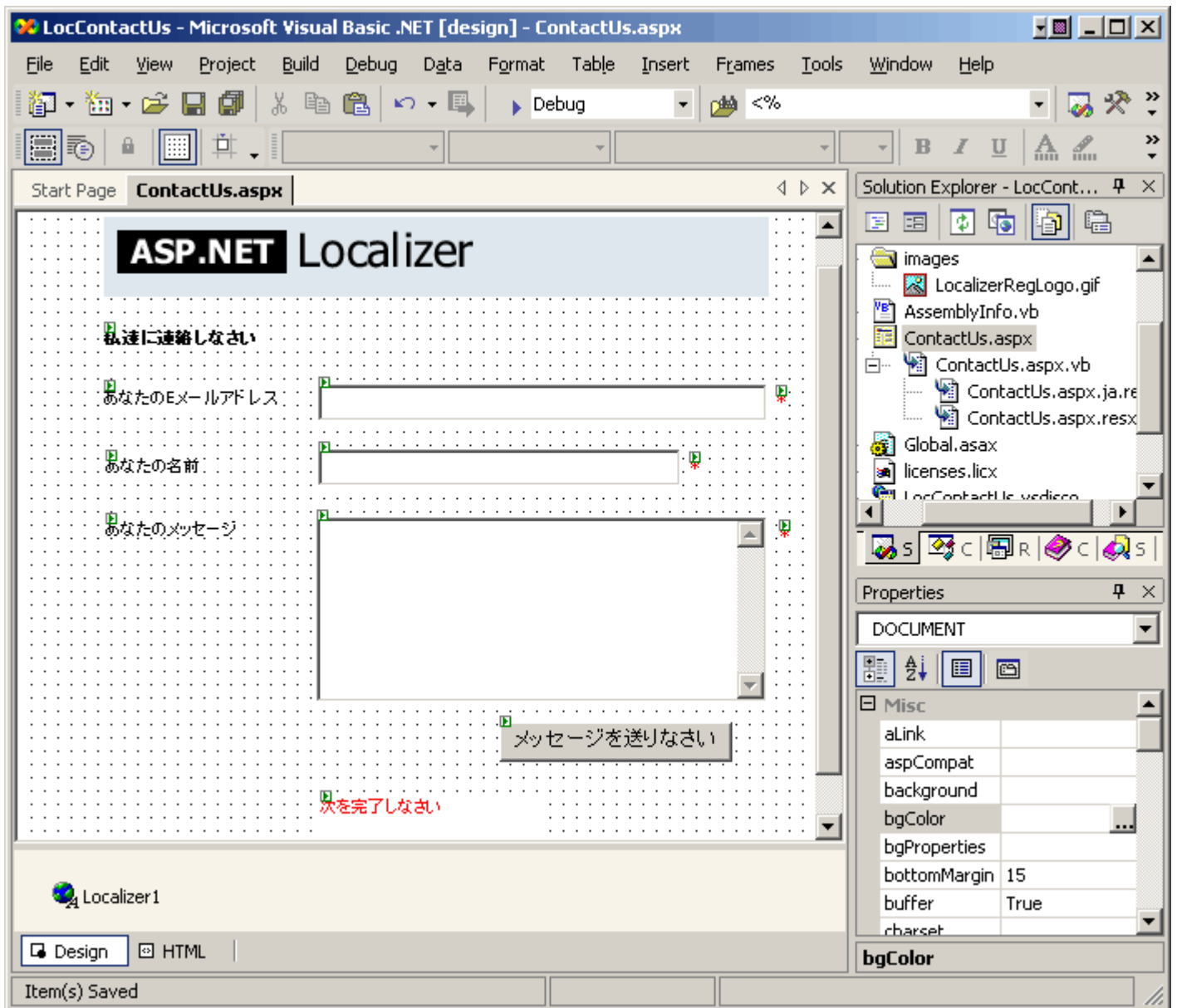
The Solution Explorer on the right shows the project structure, including the 'ContactUs.aspx.resx' file. The Properties window is also visible at the bottom right.

Now all that remains is to switch the Localizer language, and enter some localized versions of the text.

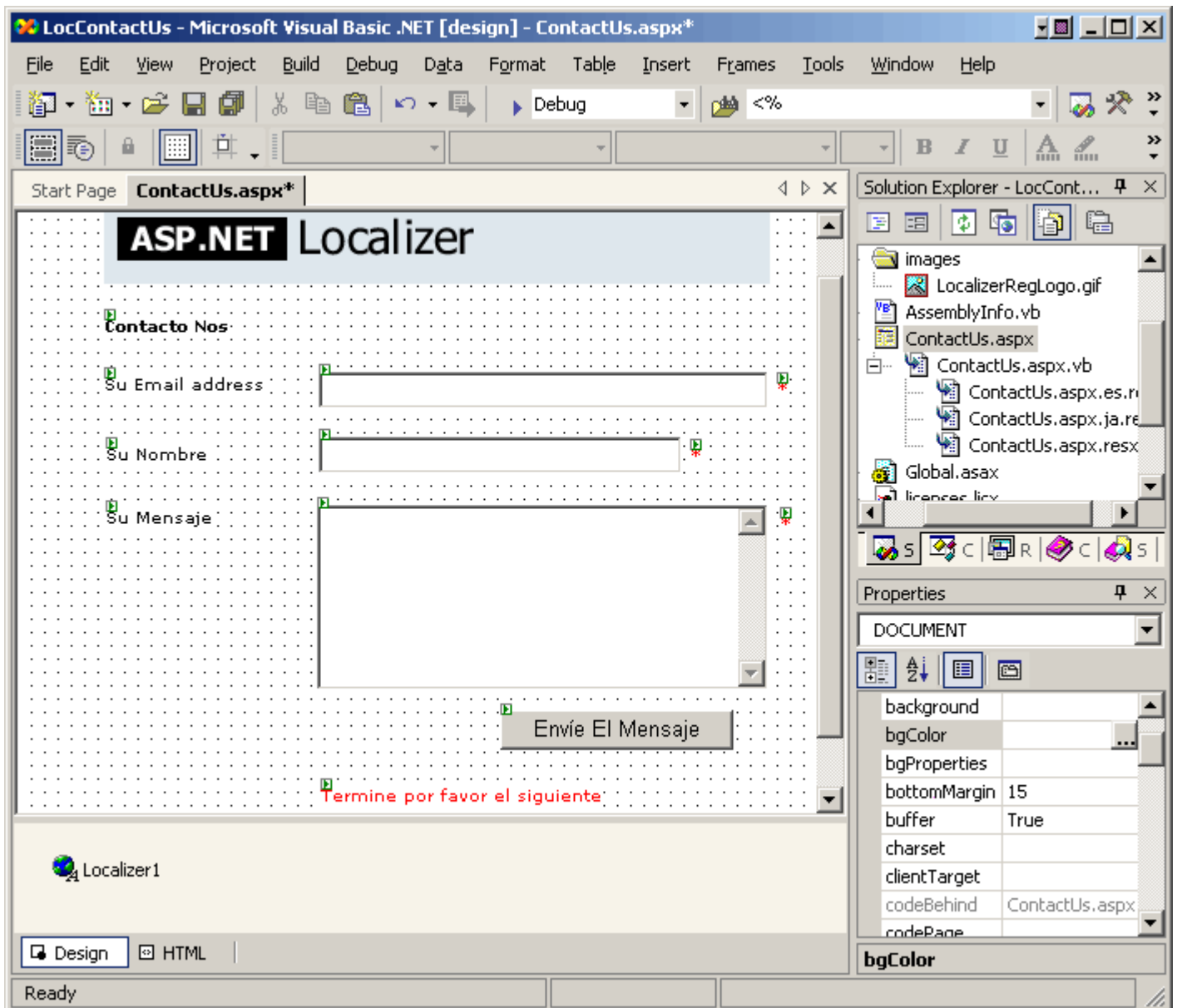
To switch language, we select the Localizer1 component in the tray and view its properties. From the Language property dropdown we select Japanese (Note how a Japanese resx file, ContactUs.aspx.ja.resx, is immediately created under ContactUs.aspx in the Solution Explorer):



Filling in the Japanese text:



And again for Spanish:

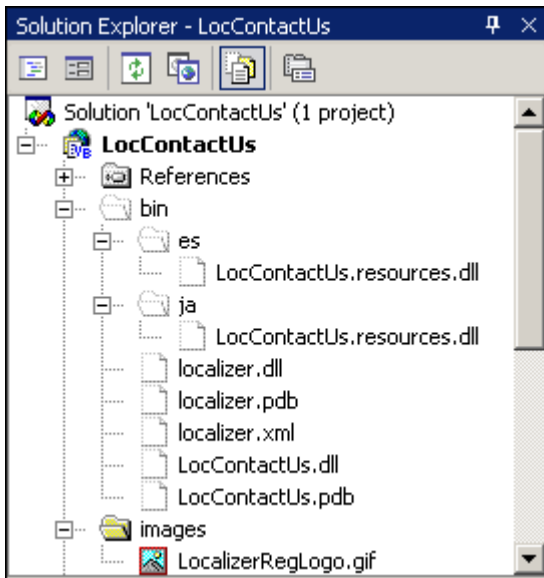


We can switch backwards and forwards between languages in the design environment, and the resx files are automatically kept up to date with any changes we make.

Note: the standard process of fallback to the Invariant culture resources always operates, i.e. if we don't supply a localized version of a property then Localizer will use the invariant resource value, both for display in the designer, and at runtime.

## Building the Project

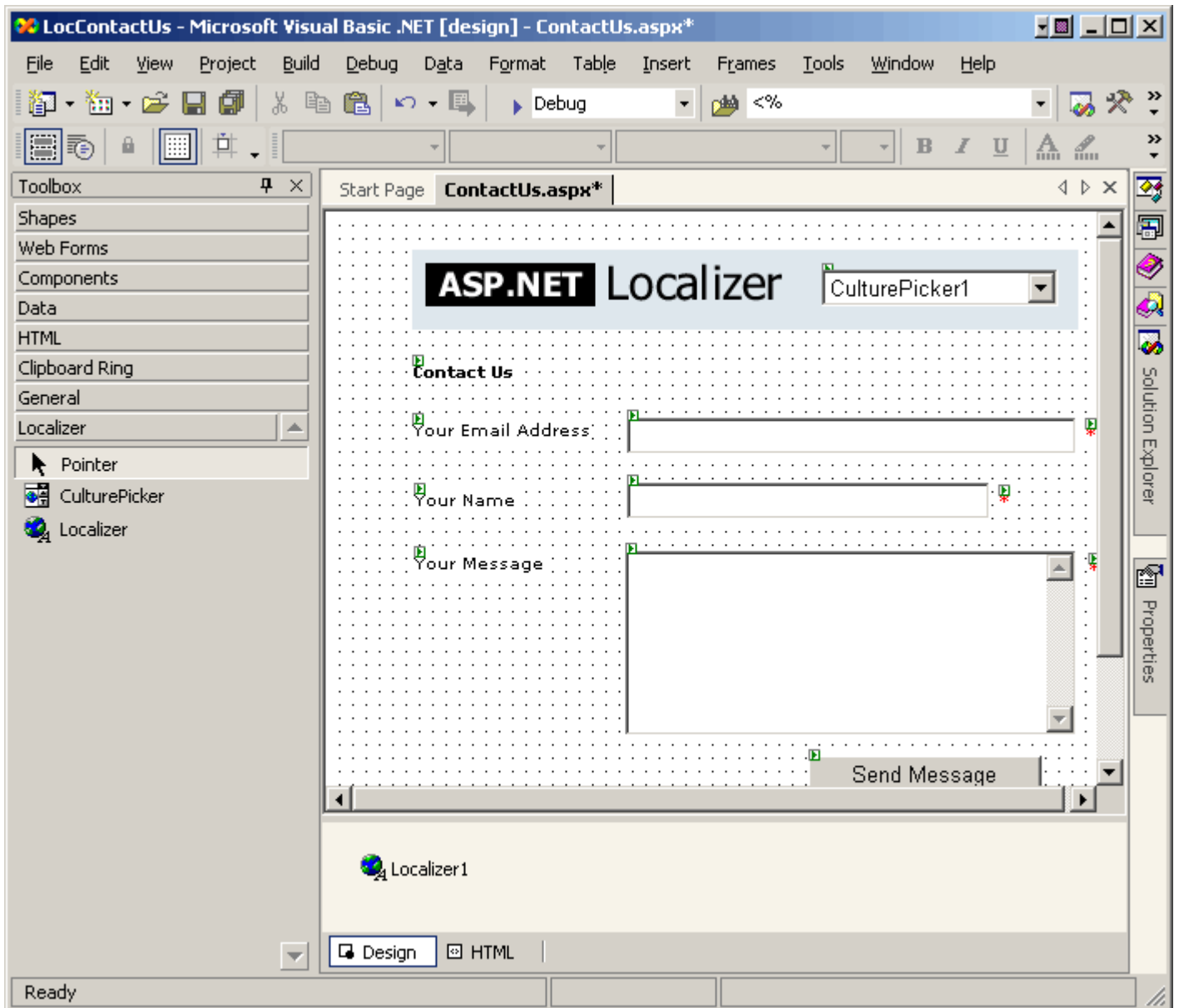
If we now build the project, we will see the satellite resource assemblies generated in their correct locations under the bin folder:



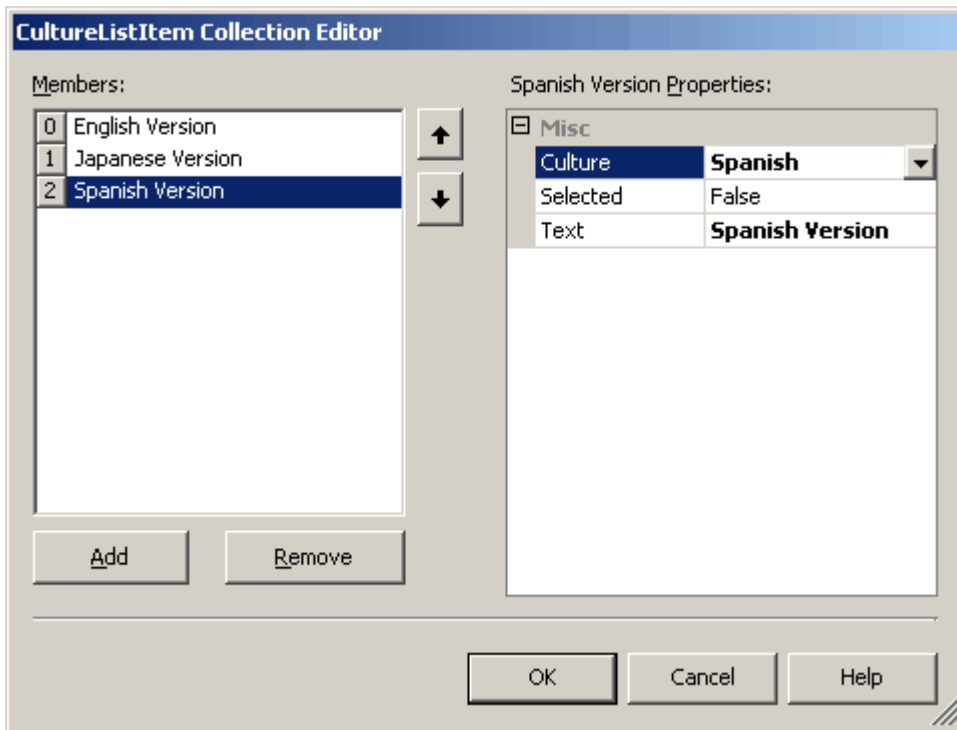
This is enough now to go on and build a localized site, all we would need to do is write code to set the thread's `CurrentUICulture` and call the `Localizer1.Localize` method.

However, instead of doing this we will use the **CulturePicker** control supplied with Localizer to prompt the user with the available languages. CulturePicker provides a design time UI for configuration, and manages the process of setting the **CurrentUICulture** at runtime.

We drop a CulturePicker onto the Form (top right hand side):



Selecting CulturePicker's Items property we add a collection item for the invariant, Japanese and Spanish cultures. Each item we specify has a text string to display and a Culture to set when it is selected. (Note that Localizer is capable of Localizing the text strings in the CulturePicker as well if required, but we won't show this here).

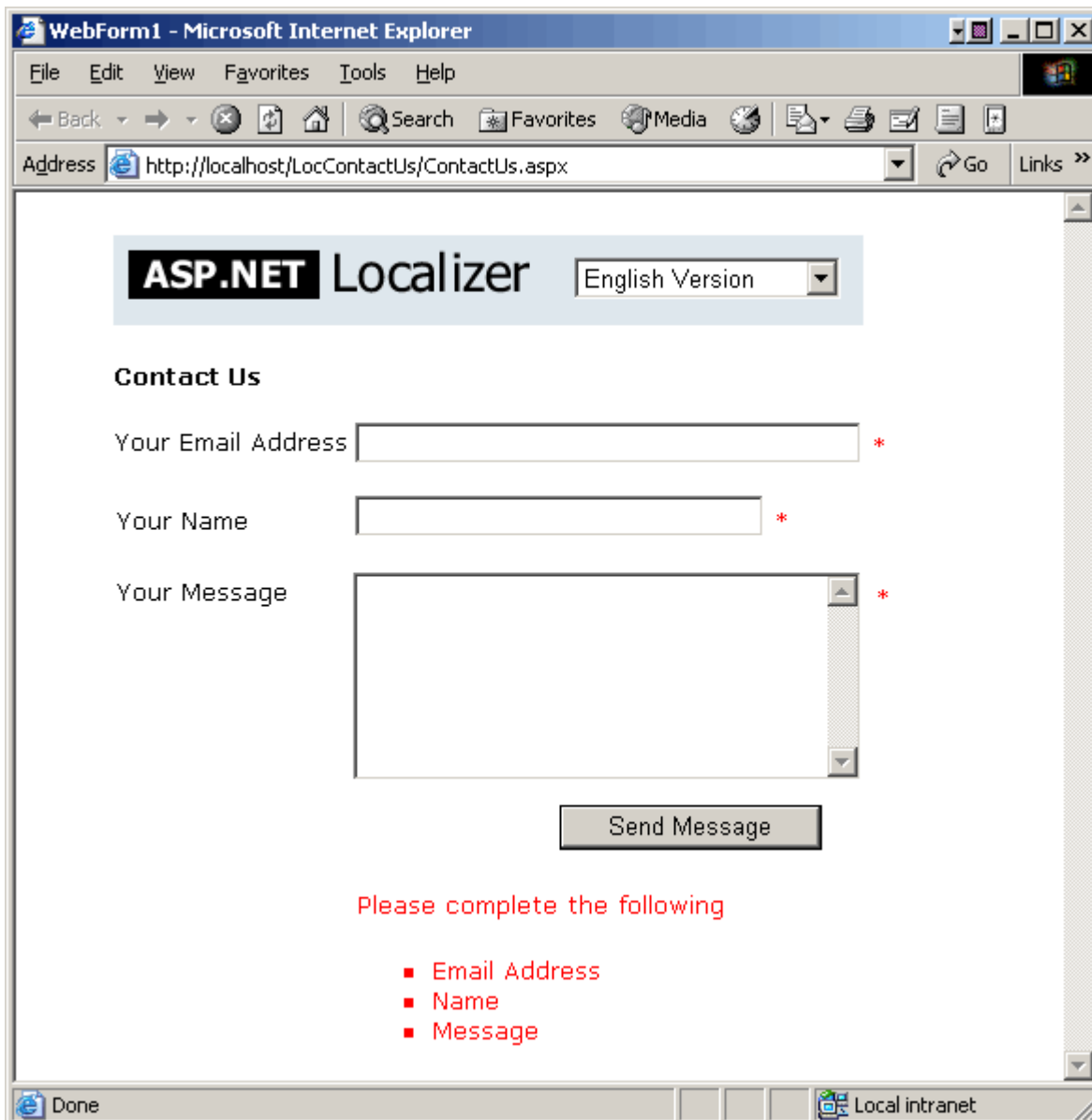


## Running the Web Application

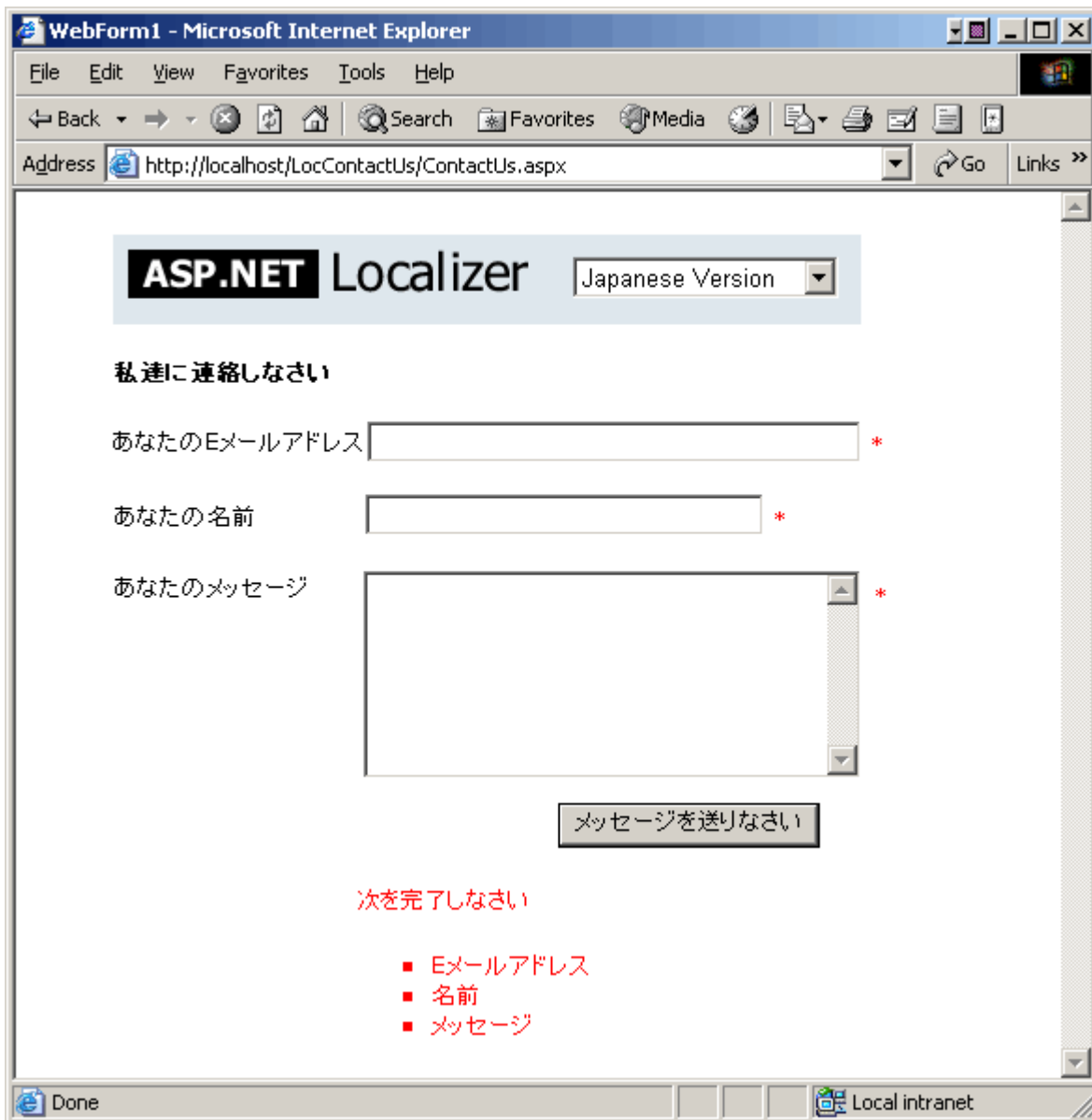
Now we've added the CulturePicker, the user has a way to select the language they want to view the page in at runtime.

The final piece in the jigsaw is to make sure that CulturePicker posts back when a selection is made. To do this set its AutoPostBack property to true. Then run the application by hitting F5 to see the results...

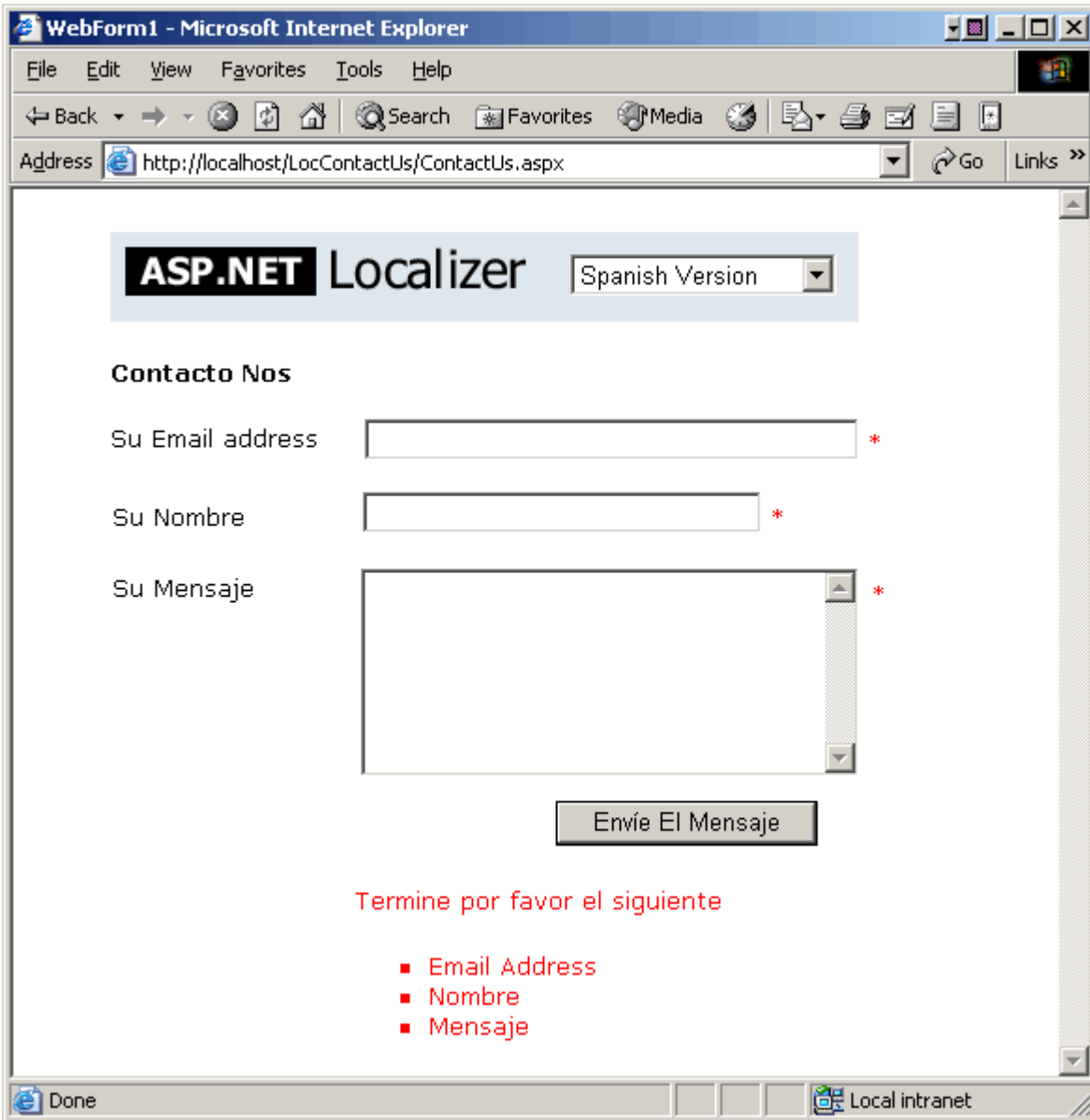
This shows the validators in action when we try and click 'Send Message' without filling in any of the required text:



Then we switch to Japanese and try again:



And finally Spanish:



## Summary

We've shown how Localizer extends the Visual Studio design time environment to provide Localization support for Web Forms, extending through to creating the resx files automatically and building the necessary satellite assemblies. Combined with the CulturePicker control for runtime language selection, Localizer provides all you need to produce a fully internationalized Web Application.

To try a free trial version of Localizer please visit the WinformReports website at <http://www.winformreports.co.uk>

June 18, 2003

© 2002-2003 [TMG Development Ltd](#)