Top 10 Tips For Internationalizing ASP.NET Applications

Guy Smith-Ferrier
guy@guysmithferrier.com
Blog: http://www.guysmithferrier.com

About…

- Author of .NET Internationalization
  - Visit http://www.dotneti18n.com to download the complete source code

- The .NET Developer Network
  - http://www.dotnetdevnet.com
  - Free user group for .NET developers, architects and IT Pros based in Bristol

Agenda

- 1. Localizing ASP.NET Web Forms
- 2. Converting HTML controls to ASP.NET controls
- 3. Setting the culture in the URL
- 4. Localizing images
- 5. Localizing JavaScript in .ASPX files
- 6. Localizing JavaScript in .JS files
- 7. Formatting strings in JavaScript
- 8. Globalizing Regular Expressions
- 9. Provider model for Strongly Typed Resources
- 10. Testing .ASPX and .ASCX files using FxCop

1. Localizing ASP.NET Web Forms

- Run Tools | Generate Local Resources

2. Converting HTML controls to ASP.NET controls

- I18N Refactorings is a free Visual Studio add-on
  - It converts HTML controls to ASP.NET controls
- It works with Visual Studio 2005 and 2008
- It converts HTML controls like this:-

```
<label id="Label1">Here is some content</label>
```

- to ASP.NET server side controls like this:-

```
<asp:Label runat="server" id="Label1">Here is some content</asp:Label>
```

2. Converting HTML controls to ASP.NET controls (continued)

- Refactor For ASP.NET
  - http://www.devexpress.com/products/net/idetools/refactorasp/
- I18N Refactorings
  - http://www.dotneti18n.com/downloads.aspx
  - http://www.codeplex.com

3. Setting the culture in the URL

- Add the following method to Global.asax:-

```
void Application_BeginRequest(object sender, EventArgs e)
{
    CultureInfo cultureInfo = GetCultureFromUrl(Request.Url);
    if (cultureInfo != null)
    {
        Thread.CurrentThread.CurrentUICulture = cultureInfo;
        Thread.CurrentThread.CurrentCulture =
            CultureInfo.CreateSpecificCulture(cultureInfo.Name);
        // strip the culture name from the front of the path
        Context.RewritePath(Request.Url.LocalPath.
            Substring(cultureInfo.Name.Length + 1));
    }
}
```

3. Setting the culture in the URL (continued)

```
private CultureInfo GetCultureFromUrl(Uri uri)
{
    string uriCultureName = uri.LocalPath.Substring(1);
    int uriIndex = uriCultureName.IndexOf('/');

    if (uriIndex == -1)
        return null;

    uriCultureName = uriCultureName.Substring(0, uriIndex);

    foreach (CultureInfo cultureInfo in
        CultureInfo.GetCultures(CultureTypes.AllCultures))
    {
        if (String.Compare(uriCultureName, cultureInfo.Name,
            StringComparison.OrdinalIgnoreCase) == 0)
            // the URI matches a culture name
            return cultureInfo;
    }
    return null;
}
```

4. Localizing images

- Set the ImageUrl in the resource entry
  - Pros:-
    - Simple
    - Control on a per image basis
  - Cons:-
    - Laborious
    - Error prone

4. Localizing images (continued)

- Add the following method to the Global.asax:-

```
void Application_BeginRequest(object sender, EventArgs e)
{
    if (Request.Url.LocalPath.ToString().ToUpperInvariant()
        .StartsWith("/IMAGES/"))
    {
        CultureInfo cultureInfo = GetCurrentUICulture();

        if (cultureInfo != null)
        {
            int index = Request.Url.LocalPath.ToString().
                ToUpperInvariant().IndexOf("/IMAGES/");

            string newUrl = Request.Url.LocalPath.ToString()
                .Insert(index + 8, cultureInfo.Name + "/");

            Context.RewritePath(newUrl);
        }
    }
}
```

4. Localizing images (continued)

```
private CultureInfo GetCurrentUICulture()
{
    if (Request.UserLanguages.GetLength(0) == 0)
        return null;

    try
    {
        return new CultureInfo(Request.UserLanguages[0]);
    }
    catch
    {
        return null;
    }
}
```

5. Localizing JavaScript in .ASPX files

- Change JavaScript like this:-

```
<script type="text/javascript">
function ShowAlert()
{
  window.alert('Hello World');
}
</script>
```

- To this:-

```
<script type="text/javascript">
function ShowAlert()
{
  window.alert('<%= GetLocalResourceObject("HelloWorld") %>');
}
</script>
```

5. Localizing JavaScript in .ASPX files (continued)

- And then to this:-

```
<script type="text/javascript">
function ShowAlert()
{
  window.alert('<%= Web.JavaScript.JSUtilities.EncodeString(
  GetLocalResourceObject("HelloWorld").ToString()) %>');
}
</script>
```

6. Localizing JavaScript in .JS files

- Change this:-

```
function ShowAlert()
{
  window.alert('Goodbye World');
}
```

- To this:-

```
function ShowAlert()
{
  window.alert(JavaScriptLibraryResources.GoodbyeWorld);
}
```

6. Localizing JavaScript in .JS files (continued)

- Add a JavaScriptLibraryResources.js file:-

```
JavaScriptLibraryResources={
"GoodbyeWorld": "Goodbye World"
}
```

- Add a JavaScriptLibraryResources.fr.js file:-

```
JavaScriptLibraryResources={
"GoodbyeWorld": "Au revoir le monde"
}
```

6. Localizing JavaScript in .JS files (continued)

- Add a reference to the script file:-

```
<script type="text/javascript" src='<%=
GetLocalResourceObject("JavaScriptLibraryResources").ToString()%>'>
</script>
```

7. Formatting strings in JavaScript

- This code uses text ("Hello") plus a parameter ("Guy"):-

```
function DisplayAlert()
{
  window.alert("Hello" + "Guy");
}
```

- Solution: Use ASP.NET AJAX's String.format:-

```
function DisplayAlert()
{
  window.alert(
      String.format('<%= GetLocalResourceObject("Hello") %>', "Guy"));
}
```

7. Formatting strings in JavaScript (continued)

- Alternatively write your own String.format:-

```
String.format = function()
{
    if( arguments.length == 0 )
        return null;

    var formatString = arguments[0];
    for(var argumentNumber = 1; argumentNumber < arguments.length;
        argumentNumber++)
    {
        var replaceExpression = new RegExp('\\{' +
            (argumentNumber - 1) + '\\}','gm');
        formatString = formatString.replace(
            replaceExpression, arguments[argumentNumber]);
    }
    return formatString;
}
```

8. Globalizing Regular Expressions

- Change this code:-

```
Regex regex = new Regex("[A-Za-z]");
if (regex.IsMatch(TextBox1.Text))
    Title = "Matches";
else
    Title = "Does not match";
```

- To this:-

```
Regex regex = new Regex(@"[\p{Lu}\p{Ll}]");
if (regex.IsMatch(TextBox1.Text))
    Title = "Matches";
else
    Title = "Does not match";
```

8. Globalizing Regular Expressions (continued)

- But JavaScript cannot use Unicode character classes so client side RegularExpressionValidators cannot use Lu and Ll
- Instead refer to the characters specifically

```
<asp:RegularExpressionValidator
   ID="RegularExpressionValidator1"
   runat="server"
   ControlToValidate="TextBox1"
   ErrorMessage="Enter letters only"
   ValidationExpression="[A-Za-z ÀÈÌÒÙ àèìòù ÁÉÍÓÚ Ý áéíóúý ÂÊÎÔÛ
   âêîôû ÃÑÕ ãñõ ÄËÏÖÜŸ äëïöüÿ ¡¿çÇŒœ ßØøÅå ÆæÞþ Ðð]*">
</asp:RegularExpressionValidator>
```

9. Provider model for Strongly Typed Resources

- The Problem:-
  - The generated code is hard wired to use ResourceManager

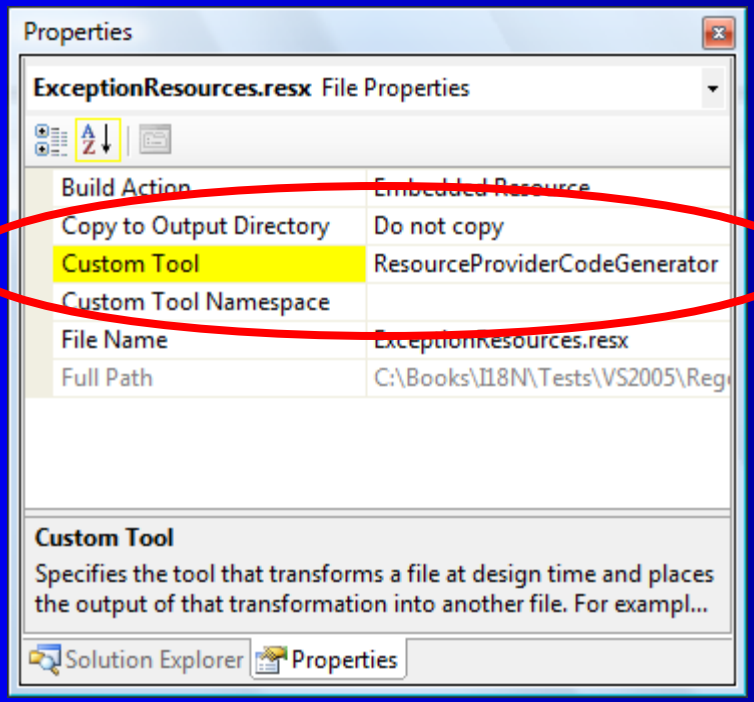
```
global::System.Resources.ResourceManager temp =
    new global::System.Resources.ResourceManager(
    "WebApplication1.ExceptionResources",
    typeof(ExceptionResources).Assembly);
```

- The Solution:-
  - Change the code to use a provider instead:-

```
System.Resources.ResourceManager temp =
    Internationalization.Resources.ResourceManagerProvider.
    GetResourceManager("WebApplication1.ExceptionResources",
    typeof(ExceptionResources).Assembly);
```

9. Provider model for Strongly Typed Resources (continued)

- To change the code permanently use a different code generator:-



10. Testing .ASPX and .ASCX files using FxCop

- FxCop is a static analysis engine for analyzing .NET assemblies
  - http://code.msdn.microsoft.com/codeanalysis
- FxCop 1.36 can analyze ASP.NET generated assemblies
  - Code from ASPX and ASCX files is placed in these assemblies
- Use a Web Deployment Project to generate these assemblies
  - Uncheck "Allow this precompiled site to be updatable"

Summary

- 1. Localizing ASP.NET Web Forms
- 2. Converting HTML controls to ASP.NET controls
- 3. Setting the culture in the URL
- 4. Localizing images
- 5. Localizing JavaScript in .ASPX files
- 6. Localizing JavaScript in .JS files
- 7. Formatting strings in JavaScript
- 8. Globalizing Regular Expressions
- 9. Provider model for Strongly Typed Resources
- 10. Testing .ASPX and .ASCX files using FxCop