Website Localization Techniques

Waseem I. Bader
Al-Salt College for Human Sciences,
Al-Balqa Applied University,
Al-Salt, Jordan

ABSTRACT
The world is becoming a smaller place with the fast communication advances that are going on, nowadays ambitious organizations are looking to upgrade their businesses from serving local regions to global ones. To achieve this internationalizing approach many linguistic and cultural differences need to be considered and website programmers and designers are faced with many technical issues. This paper will try to present different technical techniques that can be used to accomplish localization in websites.

The code samples in this paper were implemented using the ASP.net programming language, but it can be implemented in other development languages and environments using the same concepts.

Keywords
Website Localization, Website Translation, Internationalization, Multi-language website, Dynamic Website

1. INTRODUCTION
Websites nowadays are considered the most important advertising, sales, marketing, technical support or even public relation part of any company, organization or personal figure. In the ever fast growing communication era nowadays, a good website can turn any small organization serving local communities into becoming a worldwide one. [1]

This can be done through good website design, content and service, but it cannot be done without taking care of Localization.

Website localization is a term used to reflect on the process of preparing a website to have different views and content that is linguistically and culturally acceptable to different clients' from different countries or regions outside the local one. Website Localization is concerned with making the client experience with a website as local as possible to make the client feel as if this website was created just for his own country, region or even oneself! [2]

Website localization deals with translating website contents to clients' local languages, viewing the layout in the appropriate local orientation, using locally interpreted images, media or colors, correct regional date, time, currency formats, and all the aspects of cultural differences between world regions.

As much as localization is important today, it is quite a complex and hard task to achieve, that is because it has to deal with a lot of different aspects of cultural difference that need to be overcome and sometimes the cost to implement localization exceeds an organization ambition to do so.

The following figure shows some of the world languages with most native speakers including bilingual speakers, the figure clearly shows that millions of humans speak certain languages and they are completely lost as an audience to a certain organization if their language/culture is not supported in its website. [3]

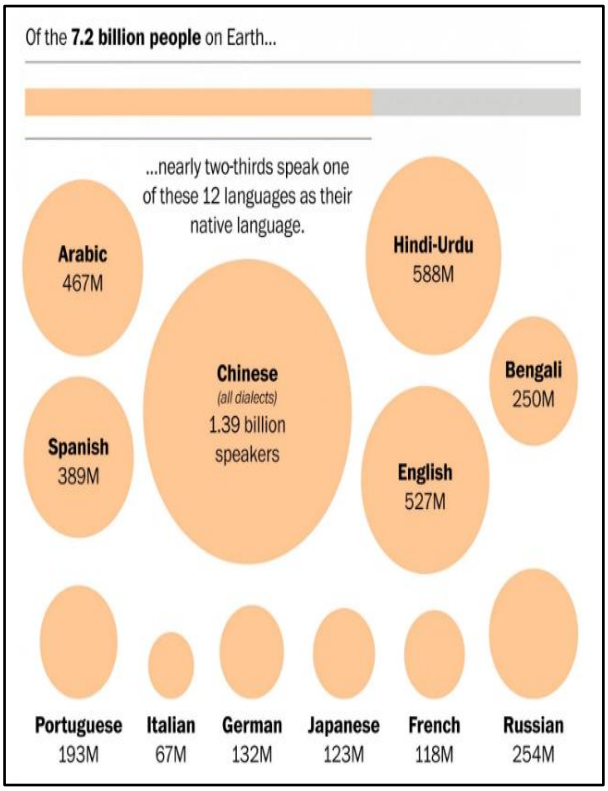

Figure 1: Some World Languages with Most Native Speakers

Websites range from simple static-content with few pages to huge portals with database-driven sites with millions of views. To localize a website first it is necessary to analyze its structure and needs and determine the clients regions of interest, and determine the differences between the various local versions. Preparing a website for localization before the development process is a key factor in its success because doing so after a website is developed is a much harder task to do. [4]

Here is an example of the Website of Al-Balqa Applied University in Jordan where the same page is viewed either in English or in Arabic based on the user preference.

Figure 2: Website Localization Example

2. LOCALIZATION ANALYSIS & PREPERATION

Localization should be prepared alongside the initial analysis phases of a website development cycle because a great deal of programming and design will be affected with the localization needs [5]. Here are some key points to take care of while preparing a website for localization:

2.1 Determine what to be localized

The first step should clearly identify what are the culturally-dependent parts of a website that need to be changed between the different versions, for example parts where text content need to be translated to different languages, or parts of the page which will have various formats based on different cultures for example numbers, dates, times, currency, addresses and telephone numbers formats, or even the orientation of the entire layout in the website. [6]

In general there are some common standards that should be taken into consideration when applying website localization these standards are published by the International Organization for Standardization. Here are some examples: [7]

- Common ISO 639: Language Codes (en = English, ar = Arabic, ch = Chinese)
- Common ISO 3166: Country Codes (us = USA, ru = Russia, cn = China)
- Common ISO 4217: Currency Codes (USD = United States of America Dollar, EGP = Egyptian Pound, JPY = Japanese Yen)
- ISO 8601: Date Format (YYYY-MM-DD = year-month-day ex. 2016-08-17)
- UPS: Universal Product Code for barcode symbology
- UPU: Universal Address Formats (numbers used in postal code to determine one's region, department, zone, sector, village)
- World Wide Web Consortium (W3C) HTML Specifications

2.2 Text Translation

Text content translation is the most integral part of website localization because in general a client is interested in the content of the website and should simply be able to understand it in his own language if possible. This step can be done by automatic translation applications or online translators but this might cause the resulting translated content to be weak or misunderstood besides many linguistic and grammatical errors are often found in their results, so using human translators with the targeted native language experience is much more useful and accurate if the organization has the ability to provide them. [8]

It is important to know that fonts of different languages need to be provided to support the characters of the targeted language and view the content properly on a localized website and the programmer of the website need to specify the correct character set encoding to be used in each version of the localized website. This is done in HTML through the use of "meta" tag "charset" attribute as shown in the following example for an Arabic content website:

```
<meta http-equiv="Content-Type"
content="text/html; charset=utf-8" />
```

Figure 1: Website Character Set in HTML

Here is a sample list of character set encodings that can be used to support different languages in a website:

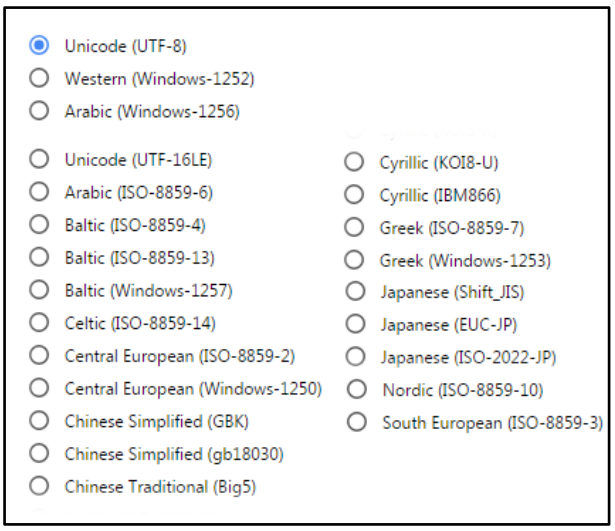

Figure 2: Sample HTML Character Set Encoding Values

2.3 Localized Symbols & Media

In an internationalized-intended website, cultural-dependent symbols should be avoided, such as icons representing local holidays, historical events or slogans instead these information can be used on the local versions of the website but not on the global one.

Images and media in general need to suit the local culture of the clients as well, for example on an international educational website, the Arabic version of the website cannot have an East Asian student representing it and vice versa.

2.4 Maintaining Localized Versions

The developed localized website should take great deal of keeping the content of all the different versions of the website up-to-date and the data should be consistent between the localized versions with no contradictions between them.

3. LOCALIZATION TECHNIQUES

In the following section different techniques that can be used to enable localization in a website will be discussed:

- Separate Localized Versions
- Resource Files
- Database Tables

3.1 Separate Localized Versions

Technique in achieving website localization is to provide separate versions of a webpage or web part for each language/culture needed.

For example, if a website needs to be in two languages (English & Arabic), there would be two separate websites or web pages for each language, each one is translated and oriented in the appropriate way with the ability to browse to the other version through hyperlinks as shown in the next figures:

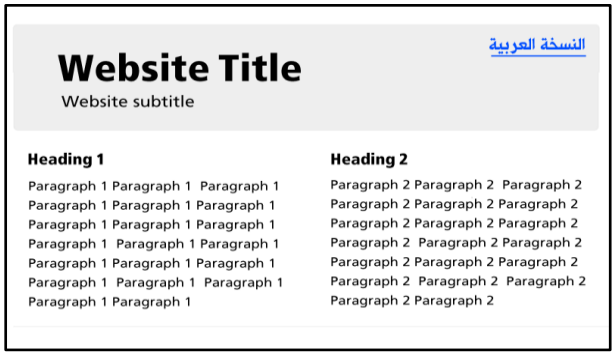

Figure 3: English Version of a Website (PageEn.html)

Figure 4: Arabic Version of a Website (PageAr.html)

Although this technique has provided some kind of localization to a website but that can only be used in some small websites with mostly static content and low need of updating, but it is considered a very bad approach to achieve website localization. Some of the many disadvantages of this technique is that it is necessary here to provide a separate website for each language targeted, and if anything in the layout of the webpage must be changed then the same change is repeated in every page for every language so the maintenance and updating of such a website will be a huge task specially for a big company or website with fast changing content or live data.

3.2 Resource Files

A resource file can be a simple text file (XML file) or a class layer that is prepared for each targeted language in a localized website, each file name must specify the language/culture that it is prepared for by using the common ISO codes mentioned before [9]. For Example, for English and Arabic versions of a website, two resource files would be prepared, if the default language is desired to be in English then the Default file name would be for example "myResources.resx", and the Arabic file name would be "myResources.ar.resx".

A separate culture for each language can also be specified by using different resource files for each language/culture pair, for example if there is a need to have different versions for Saudi Arabia and Egypt while both countries languages is Arabic, then two separate resource files are needed named "myResources.ar-sa.resx" and "myResources.ar-eg.resx", where "ar-sa" language/Country code stands for the resource file of Arabic language in Saudi Arabia Culture and the "ar-eg" stands for Arabic Language in Egypt Culture version.

These files are constructed of a resource name/value pairs and would contain actual phrases, examples, media paths, etc. for each targeted language/culture as shown in the following example:

myResource.resx

Name	Value
GeneralAlign	left
GeneralTitle	Welcome to our Website
Heading1	About us
Paragraph1	Our Company is specialized in website design and programming

myResource.ar.resx

Name	Value
GeneralAlign	right
GeneralTitle	مرحبا بكم في موقعنا
Heading1	نبذة عنا
Paragraph1	شركتنا متخصصة في تصميم وبرمجة مواقع الإنترنت

Figure 5: Resources Files Example Sample Data

In the previous example two resources files were prepared, one for the default English language "myResource.resx" and the other for the Arabic language "myResource.ar.resx". Each file contains four resources named (General Align, General Title, Heading1, Paragraph1) but each file contains different values for each resource name translated into the targeted language or has its unique value such as the case in (right/left) for the general alignment resource name. Both files are stored within the project.

Now in the desired web page, instead of using static data for titles, headings, paragraphs, etc. a reference to the resource name to be used is placed in a certain part of the webpage by using the "GetGlobalResourceObject" built-in ASP.net function. The general Syntax of the function is:

```
GetGlobalResourceObject("resource filename", "resource name")
```

Figure 6: GetGlobalResourceObject Syntax

Where in the first parameter the desired resource file name to be used is specified, in our example it would be "myResource". Notice that the language or culture part in the first parameter is not specified, but the general resource file name without the detailed localization language/culture parts is used and later the targeted file based on the page culture will be used. In the second parameter the resource name is specified in the used resource file, in the example one of the four resource names (General Align, General Title, Heading1, Paragraph1). The function will return the correct value based on the current language/culture settings and resource name. For example to get the value of the GeneralAlign resource name from myResource resource files, the syntax would be:

```
GetGlobalResourceObject("myResource",
"GeneralAlign")
```

Figure 7: GetGlobalResourceObject Example Code

Let's take a look at a sample page that would use this technique to achieve localization:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="TestResources.aspx.cs"
Inherits="Project.TestResources"%>

<html>

<body>

        <form id="form1" runat="server">


<p align="<%=
GetGlobalResourceObject("myResource","GeneralAlign")
%>">

<%= GetGlobalResourceObject(

"myResource", "GeneralTitle")%>

<br /><br />

<%= GetGlobalResourceObject(

"myResource", "Heading1")%>

<br />

<%= GetGlobalResourceObject("myResource",
"Paragraph1")%>

</p>

</form>

</body>

</html>
```

Figure 8: Resource Files Localization Example

The generated page in its default values would show the English values of the default resource file "myResource.resx" as shown in figure:

```
Welcome to our Website

About us
Our Company is specialized in website design and
programming
```

Figure 9: Resource Files Localization Example Result in Default

To change the current language "**UICulture**" attribute is added in the Page tag in the first line as following:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="TestResources.aspx.cs"
Inherits="Project.TestResources" UICulture="ar" %>

<html>
<body>
....
```

Figure 10: Changing Locale in a page

This would inform the webpage to show the content of the Arabic version of the resources files used in the page, in our example it will show the content of "myResource.ar.resx" and the resulting page will show content in Arabic as shown:

```
مرحبا بكم في موقعنا

نبذة عنا
شركتنا متخصصة في تصميم وبرمجة مواقع الإنترنت
```

Figure 11: Resource Files Localization Example Result in Arabic

To specify language/country resource file, the UICulture value is only needed to be set to the correct language and country codes as did before in naming the resource files. For example to set the UICulture to the Egyptian Arabic version of the webpage the value would be "ar-eg" instead of just "ar".

Notice that in the previous examples, the resource values are accessed in the webpage using ASP.net inline expressions which has the syntax of <%= expression %> [4] [10]. This allows us to access server-side variables or expressions within the HTML code, notice how in the previous example it is used to specify the HTML paragraph align attribute to be left oriented in the default English view and to be right oriented in the Arabic view.

This technique can also be used to assign dynamic resource values to Server-side controls by using the inline bind expressions, for example to set the same values on an ASP.net Label control in the page, the syntax would be:

```
<asp:Label ID="lblHeading1" runat="server"
Text="<%$ Resources:myResource,Heading1 %>" />
```

Figure 12: Binding a Resource to a control attribute.

Here the Text attribute will have the same value of the Heading1 resource from "myResource" based on the current Language/Culture and the same feature can be applied to other control attributes like ForeColor, BackColor, etc.

The inline Bind expression has the syntax of <%$ bind variable/reference %> and is used with server-side controls' attributes.

To change the Language/Culture for a single web page through code, the built-in "InitializeCulture" function that is called early in the page's life-cycle is overridden and in it the desired UICulture is set as shown:

```
protected void InitializeCulture()
{
    Page.Culture = "ar-SA";
    Page.UICulture = "ar-SA";
}
```

Figure 13: Changing Locale in a page through code

Notice that there are two important localization page properties used, Culture and UICulture, Culture value affects all non visual components such as Date, Time, Currency, Numbers, etc. on a page, while UICulture affects the visual elements such as the ones used in the resource files examples.

To change the Language/Culture globally for the all the pages in the entire website the general configuration file "web.config" is used by adding a globalization node under the System.web node as shown:

```
<system.web>
<globalization culture="ar-SA" uiCulture="ar-SA"/>
…
</system.web>
```

Figure 14: Changing Locale Globally

To change the Language/Culture for the all the pages in the entire website through code, the "Global.asax" file can be used, which affects all the web pages in a project, add an **"Application_BeginRequest"** function that will be called upon every request to every page, and set the correct language/culture as following:

```
public class Global : System.Web.HttpApplication
{
void Application_BeginRequest
(object sender, EventArgs e)
{
System.Threading.Thread.CurrentThread.CurrentCulture =
System.Globalization.CultureInfo.CreateSpecificCulture("ar");
System.Threading.Thread.CurrentThread.CurrentUICulture =
new System.Globalization.CultureInfo("ar");
  }
  }
```

Figure 15: Changing Locale Globally through Code

As seen this technique is quite powerful in customizing a website to view different localized views depending on the prepared language/Culture resource files. This avoids the problem of having separate websites for each language so the maintenance and updating of such websites is quite forward and simple.

Needless to say that if there is a lot of language/culture pairs to be targeted in a website, then a separate language/culture resource file need to be prepared for it, and adding a new record or removing an existing one or even changing its name will require updating that in every single file. This problem can be somehow reduced using the other techniques.

3.3 Database Tables

Localizing big fast changing websites is better performed using Databases. This technique is quite popular and suitable to avoid most of the problems faced in websites localization.

In this technique instead of using separate resource files for each language/culture as seen in the previous section, database tables are created to store all the translated contents and the different values for each language/culture in often few database tables. [11]

In this section two simple database designs will be discussed that'll be used to sell items in a multilingual website. Although both will provide localization but one design will be considered as a bad database design and the other as a good one.

The basic requirements for the database are:

- Each item will have a name, description and price.
- The design must support English and Arabic languages.

3.3.1 Database Localization - The Bad Design

This database design is used in many websites and it does provide localization but it is harder to maintain and upgrade as will be seen later, in this technique one table is usually needed with every translated value has its own column in the table as shown.

ItemsTable

Column Name	Data Type	Length
ItemId	number	7
ItemNameEn	varchar2	256
ItemNameAr	varchar2	256
ItemDescEn	varchar2	4000
ItemDescAr	varchar2	4000
ItemPrice	number	7,2

Figure 16: Items Database Table Structure - Bad Design

Every item has a unique identifier, English name, Arabic name, English description, Arabic description and price, some sample data are shown in the next figure:

ItemsTable

ItemId	ItemNameEn	ItemNameAr	ItemDescEn	ItemDescAr	ItemPrice
1	Table	طاولة	Wooden ..	خشبي ...	50.00
2	Chair	كرسي	White pl..	بلاستيك أبيض ...	25.00
3	Computer	حاسوب	New & fast	حديث وسريع ...	250.00
4	Shirt	قميص	Cotton ...	قطني ...	7.50
5	Pen	قلم	blue ...	أزرق ...	1.25

Figure 17: Items Database Table Sample Data - Bad Design

Now in the display web page, the desired content is displayed to the user by retrieving the data from the appropriate language columns in the table, if the English version is targeted, the English columns are retrieved for the item name

and description, and if the Arabic language is targeted the corresponding Arabic columns are retrieved. A sample code snippet to retrieve the desired language content is shown in the following figure:

```
String currentLanguage = ""+Session["curLang"];
DataLayer dl = new DataLayer();
DataReader itemReader = dl.getDataReader(" select
ItemId, ItemNameEn, ItemNameAr, ItemDescEn,
ItemDescAr, ItemPrice from ItemsTable ");
while (itemReader.Read())
{
String itemId;
String itemName;
String itemDesc;
String itemPrice;
itemId = ""+ itemReader["ItemId"];
itemPrice = "" + itemReader["ItemPrice"];
if (currentLanguage.Equals("en"))
{
    itemName = ""+ itemReader["ItemNameEn"];
    itemDesc = "" + itemReader["ItemDescEn"];
}
else if (currentLanguage.Equals("ar"))
{
    itemName = "" + itemReader["ItemNameAr"];
    itemDesc = "" + itemReader["ItemDescAr"];
}
//code to display item
}
da.closeConnection();
```

Figure 18: Items Server-Side Code Example - Bad Design

In the previous example, items are read from the database table through SQL and a user-defined layer called "DataLayer" is used, items identifiers and prices are read for both languages because there is no change between them, but depending on the current viewing language items' names and descriptions are chosen through a conditional statement. Notice that the targeted language is stored in the current client session and further detailed examples on using sessions for localization are explained in the next section.

Although this approach has provided localization to the current webpage, but it is considered a bad database design, because in this approach every piece of information that must be localized will need to have a separate column for each language/culture needed. So in case of huge contents and many languages expected to be provided, the required database tables will have a big complex structure and changing any column or adding a new one will be quite hard to do.

3.3.2 Database Localization - The Good Design

To overcome the disadvantages of the previous database design approach, the previous table will be split into multiple separate tables as shown in figure:

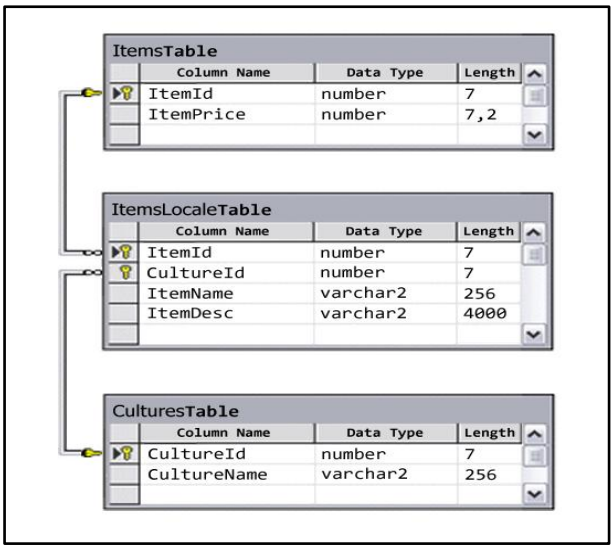

Figure 19: Items Database Table Structure - Good Design

In this design data are stored in three separate tables, some sample data for the three tables is shown in the following figure:

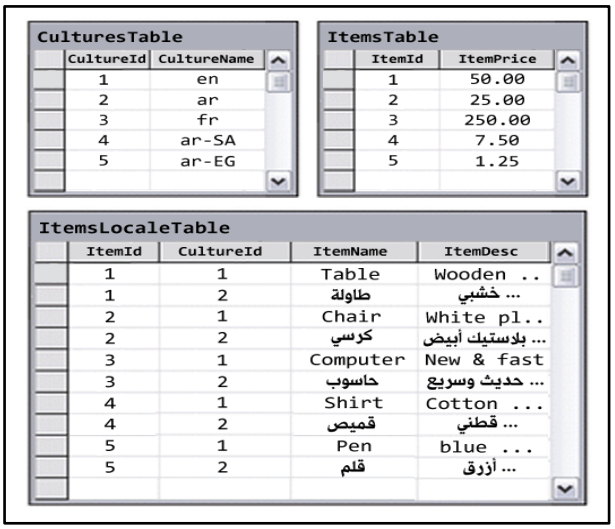

Figure 20: Items Database Table Sample Data - Good Design

The "CulturesTable" contains the supported languages/cultures in the database, The "ItemsTable" contains the items identifiers and their language-independent information in our case the price of the item, while the "ItemsLocaleTable" contains the join of the previous two tables with its language-dependent information, hence the item identifier is joined with the language/culture identifier, along with the item's name and description specific for the current culture.

The edited sample server-side code will have the SQL select statement join the three tables and select the appropriate language/culture needed. Notice that the conditional statements have been omitted because only the correct translations are returned from the SQL statement.

```
String currentLanguage = ""+Session["curLang"];
DataLayer dl = new DataLayer();
DataReader itemReader = dl.getDataReader(
"select ilt.ItemId, ilt.ItemName, ilt.ItemDesc, it.ItemPrice
from ItemsTable it, CulturesTable ct, ItemLocaleTable
ilt
where ilt.ItemId = it.ItemId
and ilt.CultureId = ct.CultureId
and ct.CultureName = '"+ currentLanguage +"' ");
while (itemReader.Read())
{
String itemId; String itemName;
String itemDesc; String itemPrice;
itemId = ""+ itemReader["ItemId"];
itemPrice = "" + itemReader["ItemPrice"];
itemName = ""+ itemReader["ItemName"];
itemDesc = "" + itemReader["ItemDesc"];
//code to display item
}  da.closeConnection();
```

This "good" localization database design approach enables us to easily maintain the database because new languages can be added to the website by only adding new records to the tables without the need to change the structure of the tables.

Using the same database design discussed here, the resources files technique can be substituted by using a similar structure, and still avoids the disadvantages of having multiple language/culture files. The next figure shows the same example used in the previous resource files section designed as a database.

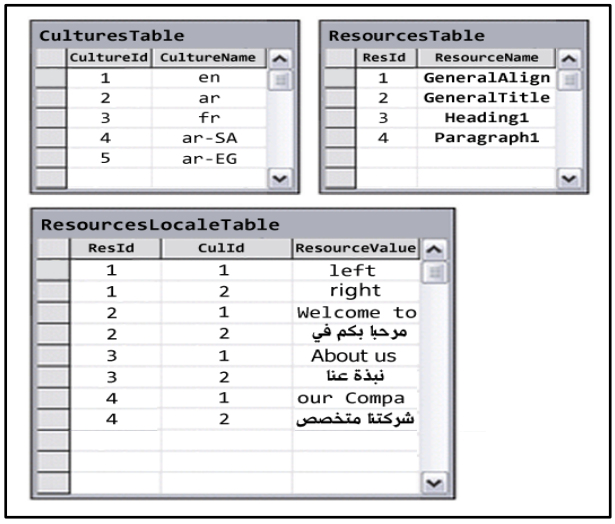

Figure 21: Database Design Substitute for Resources Files

Notice that the disadvantages discussed in using resources files are solved through this technique, by having few tables to maintain for all the different languages/cultures.

4. DYNAMIC SERVER-SIDE LOCALIZATION

In this section some programming techniques will be discussed that can be quite useful in helping website programmers and designers to provide localization capabilities with minimum maintenance effort as possible. These techniques will be discussed through code and will deal with a scenario of an English/Arabic supporting website but it can easily be upgraded to deal with any language/culture.

First of all a "LocalAccess" layer is prepared that will contain the necessary localization functionalities.

```
public class LocalAccess
{  …  }
```

Figure 22: LocalAccess Class/Layer

The First function to prepare will be called "SetLanguage" which will set the current client's language to a certain language/culture value as shown:

```
public class LocalAccess
{ ...
public static void SetLanguage(String newLang)
{
HttpContext.Current.Session.Remove("curlang");
HttpContext.Current.Session.Add("curlang", newLang);
```

Figure 23: LocalAccess SetLanguage Function

This function takes the new language/culture to be set for the current client session as the first parameter, the values will be one of the language/culture common ISO codes discussed earlier, in our example it will have "en" for English language or "ar" for Arabic language. This value will be stored in the current client's session under the name "curLang", but first any previous stored value is removed. To use this function simply call it through the class layer name as following:

```
LocalAccess.SetLanguage("ar");
//OR
LocalAccess.SetLanguage("en");
```

Figure 24: Using SetLanguage Function Example

Note that a session object is created on the web server for each client as shown in figure:

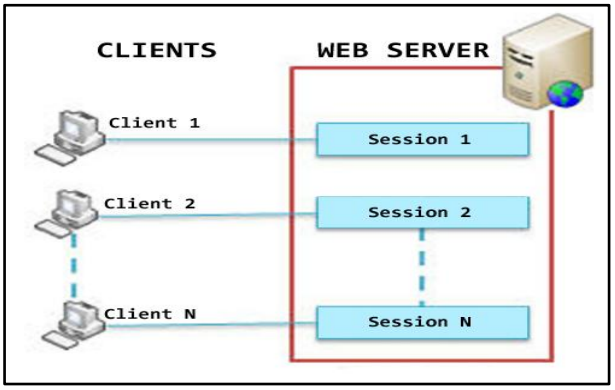

Figure 25: Session per Client on Web Server

Each session is structured as name/value pairs that a website can use to store client specific data for the current client request and is accessed through all the pages of the current web project. [12]

Session Structure

Name	Value
Name1	Value1
Name2	Value2
NameN	ValueN

Figure 26: Session Name/Value Structure

Hence the "SetLanguage" function stores the current desired language for the current client session on the server. Next we'll add a "SwitchLanguage" function to swap the current client language between English and Arabic.

```
public class LocalAccess
{
...
public static void SwitchLanguage()
{
String currentlang = "" +
HttpContext.Current.Session["curlang"];

String newlang = "";
if (currentlang.Equals("ar"))
{     newlang = "en";       }
else if (currentlang.Equals("en"))
{     newlang = "ar";       }
else       //default value
{     newlang = "en";       }
HttpContext.Current.Session.Remove("curlang");
HttpContext.Current.Session.Add("curlang", newlang);
}
…
}
```

Figure 27: LocalAccess SwitchLanguage Function

As the code implies this function will switch the current language to the "other" one without the need to explicitly set the language name as shown in figure:

```
LocalAccess.SwitchLanguage( );
```

Figure 28: Using SwitchLanguage Function Example

Now let's add two more functions in the Local layer which are "CurLangIsEnglish" and "CurLangIsArabic" which will return "True" if so and "false" otherwise.

```
public class LocalAccess
{
...
public static Boolean CurLangIsEnglish()
{
Boolean res = false;
String clang = "" +
HttpContext.Current.Session["curlang"];
if (clang.Equals("en"))
     res = true;
return res;
}

public static Boolean CurLangIsArabic()
{
Boolean res = false;
String clang = "" +
HttpContext.Current.Session["curlang"];
if (clang.Equals("ar"))
     res = true;
return res;
}
…
}
```

Figure 29: LocalAccess CurLangIsEnglish & CurLangIsArabic Functions

These functions will come very handy for the programmer to easily determine if the current client language is Arabic or English and can be directly used in conditional or iterative statements as shown in figure:

```
if ( LocalAccess.CurLangIsEnglish() )
{
//English specific code or elements
}
else if ( LocalAccess.CurLangIsArabic() )
{
//Arabic specific code or elements
}
```

Figure 30: Using CurLangIsEnglish & CurLangIsArabic Functions Example

Now a "Pick" function is prepared that will choose between two values based on the current language in use as shown in figure:

```
public class LocalAccess
{
...
public static String Pick
(String paramEnglish, String paramArabic)
{
String res = paramEnglish;
String clang = "" + HttpContext.Current.Session["curlang"];
if (clang.Equals("ar"))
{
    res = paramArabic;
}
return res;
}
}
```

Figure 31: LocalAccess Pick Function

This function will be of great help for the web programmer/designer since two values will be provided through code and the correct one will be used for the current chosen language/culture for example:

```
String siteTitle = LocalAccess.Pick("Welcome","أهلا");
```

Figure 32: LocalAccess Pick Function Example

The result of the Pick will be one of the values passed based on the current language.

We can also use the ASP.net Properties features to directly deal with language/culture orientations [10] [13], for example English language is left oriented so most of the content will be viewed on the left side of their containers, on the other hand Arabic is right oriented hence most of the content will be viewed from the right side. So using Properties can be of great help for the web design. For example a property called "LEFT" is created, this property will mean left in the English version of the website but will automatically mean right for the Arabic one as shown in figure:

```
public class LocalAccess
{
public static String LEFT
{
get
{
String res = "left";

String clang = "" + HttpContext.Current.Session["curlang"];

if (clang.Equals("ar"))
{
    res = "right";
}

return res;
}
}
}
```

Figure 33: LocalAccess LEFT Property

This property will return "left" in English version and "right" for the Arabic one, this approach will come very handy in designing localized HTML layouts as following:

```
<p align="<%= LocalAccess.LEFT %>">
    //Paragraph content here
</p>
```

Figure 34: LocalAccess LEFT Property Example

As the example shows, this property will automatically have different values based on the current language. And the layout of the whole paragraph will change dynamically through code without the need for the web design to explicitly write all the needed code every time a language/culture orientation or language-dependent design value is used. Likewise similar Properties can be prepared for "RIGHT" and others for direction right-to-left "RTL" and left-to-right "LTR", etc…

5. CONCLUSION

World Wide Web really means serving a wide range of cultures and regions in the world, so today localization is an integral part of any ambitious organization.

Having a site with multiple languages and cultures is easily said than done, because having different localized version of a website will need more maintenance, their content should be consistent and language/cultural dependent data should be translated to suit every version.

In this paper some different techniques to achieve website localization have been discussed in details showing their advantages and disadvantages. These techniques are presented in order to help website programmers and designers to understand them and how to implement them in their work. Further researches on this important topic is expected and needed to further improve the performance and abilities of achieving better website localization.

6. REFERENCES

[1] Y. Lee and K. Kozar, "Investigating the effect of website quality on e-business success: An analytic hierarchy process (AHP) approach," School of Business, University of Kansas, United States, 2005.

[2] P. Sandrini, "Website Localization and Translation," in MuTra 2005 – Challenges of Multidimensional Translation: Conference Proceedings, Saarbrücken, Germany, 2005.

[3] R. Noack and L. Gamio, "Ulrich Ammon, Dusseldorf University – Population Reference Bureau," Independent Newspaper, 31 December 2015. [Online]. Available: http://www.independent.co.uk/news/world/the-worlds-languages-in-seven-maps-and-charts-a6791871.html. [Accessed 18 August 2016].

[4] M. A. Jimenez-Crespo, Translation and Web Localization, Oxford, United Kingdom: Routledge, 2013.

[5] S. Maheshwari and D. C. Jain, "A Comparative Analysis of Different types of Models in Software Development Life Cycle," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 2, no. 5, 2012.

[6] O. D. Troyer and S. Casteleyn, "Designing Localized Web Sites," The Web & Information Systems Engineering (WISE) Laboratory, Brussel, Belgium, 2014.

[7] J. Maroto and M. De Bortoli, "Web Site Localization," in Crossing Cultures, London, United Kingdom, 2003.

[8] F. Costales, "Translation 2.0. The localization of institutional websites," CETRA Research Seminar in Translation Studies, Leuven, Belgium, 2008.

[9] "ASP.NET Web Page Resources Overview," Microsoft, [Online]. Available: https://msdn.microsoft.com/en-us/library/ms227427.aspx. [Accessed 18 August 2016].

[10] A.R. Jones, Mastering ASP.NET with Visual C#, Indiana, United States of America: John Wiley & Sons, 2006.

[11] Yu, "Method and system for generalized localization of electronic documents". California, United States of America Patent US 2004/0205118 A1, 14 October 2004.

[12] Evjen, Professional ASP.NET 2.0, Indiana, United States of America: John Wiley & Sons, 2006.

[13] J. Liberty and D. Hurwitz, Programming ASP.NET, 3rd ed., California, United Stated of America: O'Reilly Media, 2006.