Migration of desktop applications to the web using Visual WebGui

Introduction
The following document will describe the various migration options of desktop/smart client applications to the web available using Visual WebGui.

In order to discuss the migration process of legacy desktop applications to the web, we should first agree on 3 different types of desktop applications:

1. **WinForms based desktop application** (C#/VB.NET).
   The UI layer is coded using .NET languages – the business can be .NET, COM+ or any other interop.

2. **VB 6.0 based applications**.
   The UI layer is coded with VB 6.0.

3. **Other desktop or smart client technologies** (C++ MFC/ATL, Delphi, Java etc).
   Any other smart client technology based applications.

WinForms based desktop applications

Background
Normally, without Visual WebGui the migration process of a WinForms desktop application to the web will require a full re-engineering of the UI layer to fit the web architecture and capabilities.

If we take WinForms migration to ASP.NET for example using any AJAX 3rd party controls in order to provide a rich UI experience, we will have to consider:

4. Entirely new API.
5. Entirely new update approach.
6. Entirely new look & feel – or work hard to customize the UI to look the same.
7. Lighten the amount of data transferred to the client and presented at any given time to avoid severe latency.
8. Compromise on features list due to the web limitations.
9. Handling security holes created as a result of opening services consumed by client side AJAX and transferring business logics to the client.

Visual WebGui SDK is fully integrated with Visual Studio and provides the exact same API and set of tools/capabilities which are provided out-of-the-box with WinForms 1.0 and 2.0. This fact enables the native capability of simply copying any existing WinForms source code to a VWG project and providing a fully functional equivalent web application.

The Process
The basic 3 steps of migration (**view a walkthrough tutorial**):

1. Open a new Visual WebGui application.
2. Copy the code from your WinForms project into this new web application.

Any standard WinForms application which is using the 58 WinForms out-of-the-box controls will then compile and execute as a plain web application.
Visual WebGui - Power for rich .NET AJAX applications

The outcome of this short process is an ASP.NET based solution in terms of deployment and runtime and has the following properties:

A. **Deployment** is copy & paste equivalent to an ASP.NET web site.
B. **Server infrastructure** requires an IIS and .NET CLR only.
C. The application can be consumed from any plain browser - no installation is made on the client.
D. Minor static and cached footprint on the client ~200kb of plain JS and HTML code due to the Empty Client concept.
E. Support for multiple presentation layers with the same code base (DHTML/Silverlight or Smart Client)
F. Maintain a single layer of code – no need to write or maintain JavaScript, HTML and services
G. Highly secured due to the Empty Client concept.

**Considerations & Exceptions**

There are 3 major setbacks you might have on the process which you can quantify in advance and estimate the amount of work that has to be done in order to migrate your application:

1. ** Minor differences between the VWG API and WinForms** which are mainly caused by architecture differences.
2. The amount of 3rd party controls that are used in your application.
   This section describes a situation of using some non-WinForms out-of-the-box controls (for example Infragistics or DevExpress controls etc). In those cases you can select the most suitable solution from the following 3 options:
   - a. Choose a similar control from the WinForms out-of-the-box, adjust your code to use it and then perform the migration process.
   - b. Select an equivalent 3rd party ASP.NET control (Infragistics, Telerik, DevExpress etc.) which provides the same functionality, wrap it by a click of a button in VWG and adjust your code to use it.
   - c. Write your own VWG custom control which will perfectly suit your needs and then adjust your code after the migration process to use this control.
3. Adjustments of a single user desktop application to a multiuser web environment. This section summarizes some of the major issues of transforming a single user application to a multiuser application sharing the same CPU, Memory space and other resources.
   - d. **Thread safety** – since a WinForms application can contain static members which are accessible to a single user, you should now consider one of the following:
     - i. Replacing those static members to a synchronized multi-thread safe data structures.
     - ii. Lock critical read/write sections to protect concurrent multi user access.
     - iii. Remove the statics and find instance or DB based solutions.
   - e. **Memory load** – in a desktop application, there might be places when the original consideration of the amount of memory in use was based on the assumption that the executing machine is local. Therefore, many items are loaded to memory simultaneously without limitation. Now, on a shared memory environment, when the server does the heavy lifting, the amount of memory consumed by each user will set the number of concurrent users that can be served by each server.

   The following steps are recommended:
Visual WebGui - Power for rich .NET AJAX applications

i. Consider loading items to memory on demand (keep only the headers and the identifiers in memory).
ii. Remove any large objects read to memory – for example, don’t save binary objects to memory, instead write the binary to the response stream directly to the client.
iii. Prefer DB based paging on entire prefaces and memory based paging. Visual WebGui provides mechanisms to enable it easily.

Summary
Migration of any WinForms application to the web using Visual WebGui has the following advantages:

1. In 3 simple steps you will be able to get very close to a working web application.
2. The effort you have to make in order to accomplish a fully functional web application is measurable.
3. The application can keep using the existing BL and DL layers, and only the UI is either migrated automatically or adjusted.

VB 6.0 Based desktop applications

Background
Since the WinForms development paradigm and API are actually an evolution of VB 6.0, migration to Visual WebGui which is using the exact same paradigm and API is most natural.

The Process
VB 6.0 based desktop applications can be transferred to web using Visual WebGui in 3 different approaches:

1. VB 6.0 Project → WinForms .NET Project → Visual WebGui .NET Project
   
   Migrate the VB 6.0 using Artinsoft tools and services to a WinForms desktop application and then follow the process of WinForms applications migration as described in the first section of this document.

2. VB 6.0 UI using VB 6.0 DL/BL → Visual WebGui UI using VB 6.0 DL/BL
   
   The application will keep using the existing BL/DL layers as COM/COM+ VB 6.0 based components and only the UI is mimicked within a Visual WebGui project using the productive proven tools of WinForms.
   The VB 6.0 to WinForms wizard of Artinsoft which exists within Visual Studio to make the process partially automatic.

3. VB 6.0 Project → Visual WebGui Project
   
   The entire application is mimicked within a Visual WebGui project and .NET using the productive proven tools of WinForms.
   The VB 6.0 to WinForms wizard of Artinsoft which exists within Visual Studio to make the process partially automatic.

Considerations & Exceptions
Except for the considerations and exceptions mentioned in the previous section “WinForms based desktop applications”, there are two additional subjects derived from the VB history of the project:

1. The source project written in VB 6.0 and might use non-Thread-Safe win32 or native API resources – those should be handled as potentially dangerous to the multi user web environment.

2. Threading Model: activating a COM/COM+ components coded in VB 6.0 is using “Single Thread” or “Single Thread Apartment” threading models; Consider using the “STAThread” attribute decoration above any method that is accessing VB 6.0 based COM/COM+.
Visual WebGui - Power for rich .NET AJAX applications

Summery
Migration of VB 6.0 application to the web using Visual WebGui has the following advantages:

1. The application can keep using the existing BL/DL services – no re-write or re-engineering is required.
2. UI concepts, design patterns and practices can be migrated as is.
3. UI elements and API in VB 6.0 are very similar to WinForms so that migration of the UI to Visual WebGui is quite native.
4. The effort you have to make in order to accomplish a fully functional web application is measurable.

Other desktop or smart client technologies

Background
Migration of any other desktop technology such as C++ MFC, Delphi or Java Smart Client is quite complicated to accomplish with the existing tools and without Visual WebGui.

Visual WebGui provides the shortest path for such migration needs since it maintains the entire desktop concepts including development patterns, design patterns such as MVC and other desktop UI oriented directions.

The Process
Desktop applications can be transferred to web using Visual WebGui in 2 different approaches:

1. Source technology ➔ Visual WebGui UI using DL/BL interop to Source Technology

   The application will keep using the existing BL/DL layers as interop to C++ dlls, Delphi COM Exposed API (Technical Article) or a .NET to Java bridge (Starting Point Article) and only the UI is mimicked within a Visual WebGui project using the productive proven tools of WinForms.

2. Source technology ➔ Visual WebGui Project

   The entire application is mimicked within a Visual WebGui project and .NET using the productive proven tools of WinForms.

   Gizmox has performed such experiment of migration; the “WebMail Project” is a web copy of the desktop version of Outlook 2007 (read more on WebMail project...).

Considerations & Exceptions
Except for the considerations and exceptions mentioned in the first section “WinForms based desktop applications”, there are two additional subjects derived from the VB history of the project:

1. The source desktop project coded on the source technology might use non-Thread-Safe win32 or native language API resources – those should be handled as potentially dangerous to the multi user web environment.

2. Threading Model: Note that when interoperating or bridging components coded in a different technology there might be a difference in threading model; Consider using the “STAThread” attribute decoration above any method that is accessing VB 6.0 based COM/COM+.

Summery
Migration of VB 6.0 application to the web using Visual WebGui has the following advantages:

1. The application can keep using the existing BL/DL services – no write or re-engineering is required.
2. UI concepts, design patterns and practices can be migrated as is.
Visual WebGui - Power for rich .NET AJAX applications

3. Smart Client UI elements, API and behaviors are covered by the WinForms out-of-the-box so that migration of the UI to Visual WebGui is quite native.
4. The effort you have to make in order to accomplish a fully functional web application is measurable.