Traditional Ajax vs. New business targeted Ajax

By Itzik Spitzen

Introduction

This article compares between the traditional Ajax represented by ASP.NET to that of the new Ajax approach represented by Visual WebGui. The article present an opportunity for developers to learn a way to focus their development efforts on algorithms, requirements and business logics using the new approach to Ajax which provides maximal flexibility, interoperability and interactivity with any traditional web applications, controls set and architectures.

Visual WebGui – is the platform that represents the New Ajax approach suggested by this document.

ASP.NET – is referred to, in this document, as a point of comparison, although, any other traditional Ajax development paradigm could be chosen. In other words, this document compares a more generic approach the traditional Ajax development approach with the Visual WebGui's new generation Ajax approach.

Developer considerations

We will dive into the following developer perspectives:

1. Development languages & environment
2. Required skills
3. Development effort and ease
4. API quality and convenience
5. Ability to use common and proven design patterns
6. Debugging & testing efforts needed
7. Technologies contemporariness

Choosing an infrastructure

In addition, we will highlight some major issues to examine before you go about choosing a platform to base your next line-of-business or data centric application on. The following subjects will be briefly discussed in this document:

1. Infrastructure standards

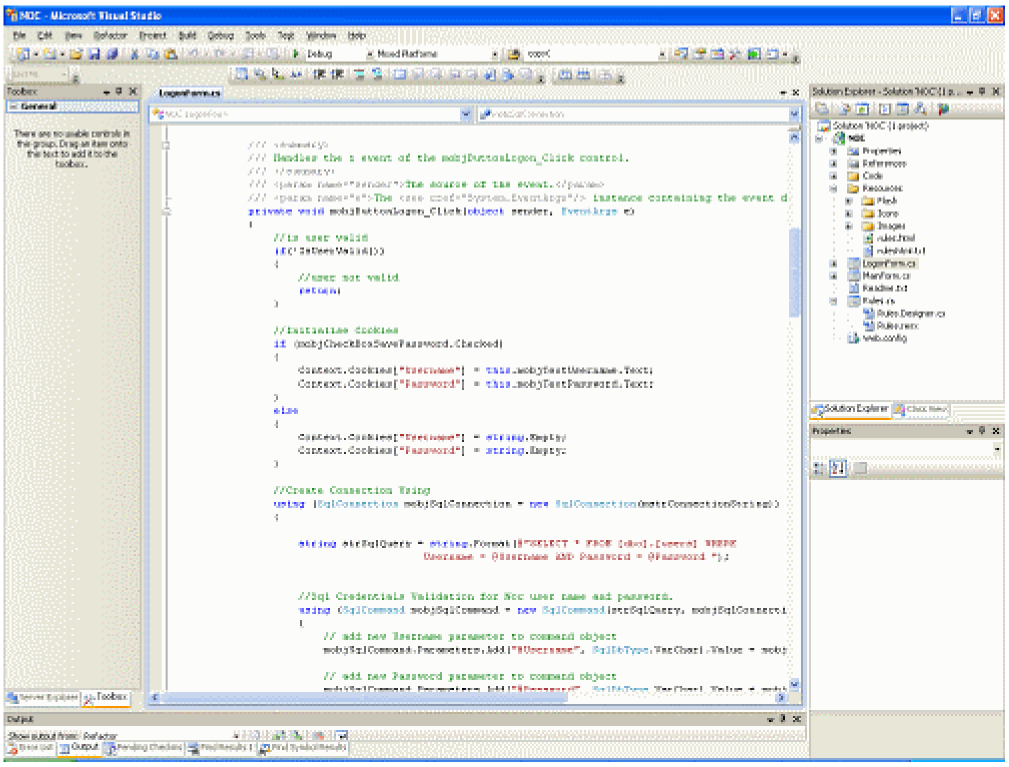
Developer considerations

This section explores some major developer's considerations when choosing a development platform to base the next project on.

Development Languages & Environment

You will probably choose your development language according to your existing skills and the skills that are most common on your organization.

Visual WebGui developers can choose any .NET language to develop applications with (i.e. C#, VB .NET etc). The Visual WebGui SDK is fully integrated into all of the Visual Studio versions (figure 1).

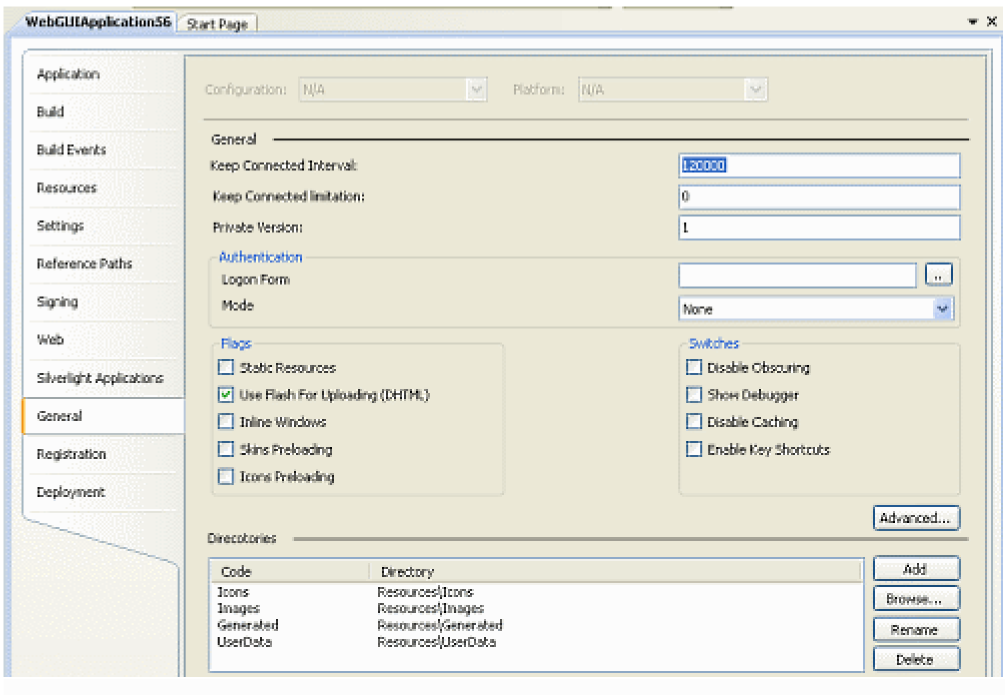


```
/// <summary>
/// Handles the : event of the mobjButtonLogon_Click control.
/// </summary>
/// <param name="sender">The source of the event.</param>
/// <param name="e">The <see cref="System.EventArgs"/> instance containing the event d
private void mobjButtonLogon_Click(object sender, EventArgs e)
{
    //is user valid
    if(!IsUserValid())
    {
        //user not valid
        return;
    }

    //Initialize Cookies
    if (mobjCheckBoxSavePassword.Checked)
    {
        Context.Cookies["Username"] = this.mobjTextUsername.Text;
        Context.Cookies["Password"] = this.mobjTextPassword.Text;
    }
    else
    {
        Context.Cookies["Username"] = string.Empty;
        Context.Cookies["Password"] = string.Empty;
    }

    //Create Connection String
    using (SqlConnection mobjSqlConnection = new SqlConnection(mstrConnectionString))
    {

        string strSqlQuery = string.Format(@"SELECT * FROM [dbo].[users] WHERE
                             Username = @Username AND Password = @Password ");


        //Sql Credentials Validation for Noc user name and password.
        using (SqlCommand mobjSqlCommand = new SqlCommand(strSqlQuery, mobjSqlConnecti
        {
            // add new Username parameter to command object
            mobjSqlCommand.Parameters.Add("@Username", SqlDbType.VarChar).Value = mobj

            // add new Password parameter to command object
            mobjSqlCommand.Parameters.Add("@Password", SqlDbType.VarChar).Value = mobj
```

Fig. 1: Visual WebGui – Native .NET development within Visual Studio

Visual WebGui adds user interfaces to simplify the application's configuration (figure 2).

Using ASP.NET without Visual WebGui require manual, xml based configuration (figure 3).



Configuration: N/A Platform: N/A

General
Keep Connected Interval: 180000
Keep Connected limitation: 0
Private Version: 1

Authentication
Logon Form
Mode: None

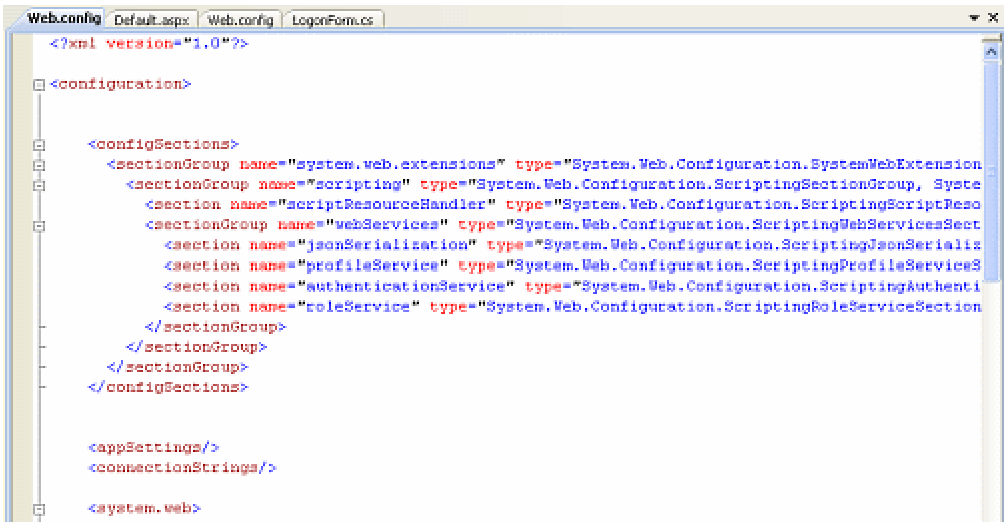
Flags
- [ ] Static Resources
- [x] Use Flash For Uploading (DHTML)
- [ ] Inline Windows
- [ ] Skins Preloading
- [ ] Icons Preloading

Switches
- [ ] Disable Obscuring
- [ ] Show Debugger
- [ ] Disable Caching
- [ ] Enable Key Shortcuts

Directories

Code	Directory
Icons	Resources\Icons
Images	Resources\Images
Generated	Resources\Generated
UserData	Resources\UserData

Fig. 2: Visual WebGui's Configuration UI

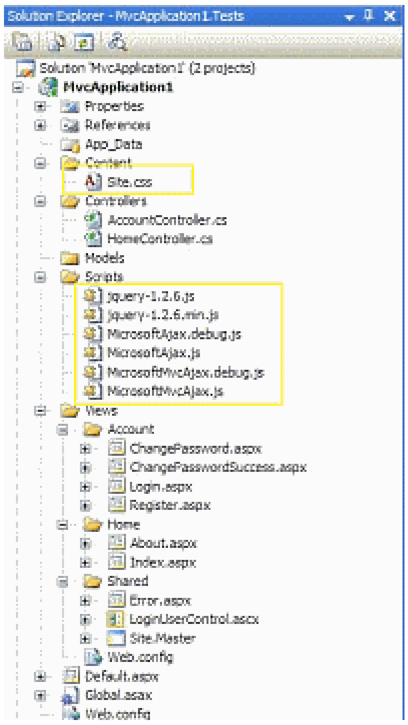


```
<?xml version="1.0"?>

<configuration>


    <configSections>
      <sectionGroup name="system.web.extensions" type="System.Web.Configuration.SystemWebExtension
        <sectionGroup name="scripting" type="System.Web.Configuration.ScriptingSectionGroup, Syste
          <section name="scriptResourceHandler" type="System.Web.Configuration.ScriptingScriptReso
          <sectionGroup name="webServices" type="System.Web.Configuration.ScriptingWebServicesSect
            <section name="jsonSerialization" type="System.Web.Configuration.ScriptingJsonSerializ
            <section name="profileService" type="System.Web.Configuration.ScriptingProfileServiceS
            <section name="authenticationService" type="System.Web.Configuration.ScriptingAuthenti
            <section name="roleService" type="System.Web.Configuration.ScriptingRoleServiceSection
          </sectionGroup>
        </sectionGroup>
      </sectionGroup>
    </configSections>


    <appSettings/>
    <connectionStrings/>

    <system.web>
```

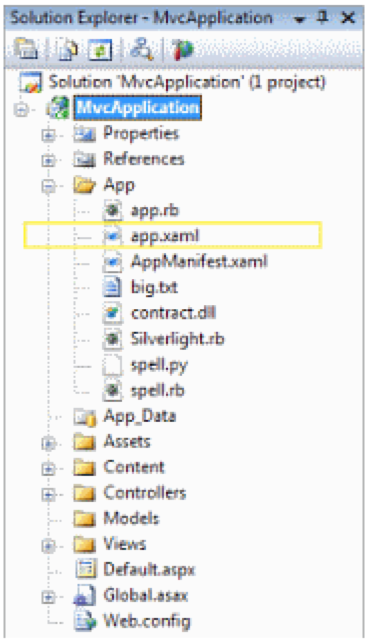
Fig. 3: ASP.NET XML Web Configuration

Required Skills

If you have used ASP.NET as well as any other traditional web development paradigms, you already know that any application development task will require additional client side UI/styling languages and concepts: JavaScript & DHTML, CSS, Silverlight, Flex/Flash, Java Applets, and ActiveX etc. (figure 4 and 5).



```
Solution Explorer - MvcApplication1.Tests
Solution 'MvcApplication1' (2 projects)
  MvcApplication1
    Properties
    References
    App_Data
    Content
      Site.css
    Controllers
      AccountController.cs
      HomeController.cs
    Models
    Scripts
      jquery-1.2.6.js
      jquery-1.2.6.min.js
      MicrosoftAjax.debug.js
      MicrosoftAjax.js
      MicrosoftMvcAjax.debug.js
      MicrosoftMvcAjax.js
    Views
      Account
        ChangePassword.aspx
        ChangePasswordSuccess.aspx
        Login.aspx
        Register.aspx
      Home
        About.aspx
        Index.aspx
      Shared
        Error.aspx
        LoginUserControl.ascx
        Site.Master
      Web.config
    Default.aspx
    Global.asax
    Web.config
```



```
Solution Explorer - MvcApplication
Solution 'MvcApplication' (1 project)
  MvcApplication
    Properties
    References
    App
      app.rb
      app.xaml
      AppManifest.xaml
      big.txt
      contract.dll
      Silverlight.rb
      spell.py
      spell.rb
    App_Data
    Assets
    Content
    Controllers
    Models
    Views
    Default.aspx
    Global.asax
    Web.config
```

Fig. 4: ASP.NET MVC Application   Fig. 5: Silverlight MVC Application

Visual WebGui abstracts away from traditional web development complexities and provides single layered .NET server development platform.

Development effort and ease

Plain ASP.NET development provides a generic environment for web sites development, however, when it comes to applications development, and mainly data-centric, line of business ones, the concepts of pages, post-backs, individual HttpXml requests and multi-layered client languages are forcing you to handle more infrastructural questions then application behavior questions.

Visual WebGui offers you to upgrade your data-centric and line of business applications development experience to the level of Forms, Events, Dialogs, Event driven Inter-controls communications and intuitive object oriented behaviors.

Let's take a look at a simple task as opening a data bound modal dialog window, a very common task especially for line of business web applications:

Standard web client code

```
function PopupWindow(url) {
var ie=document.all;
if (ie)
{
       window.showModalDialog(url,"Window","status:no; help:no;
                        dialogWidth:1200px; dialogHeight:768px");
}
else {
     var mywindow =window.open( url, "Popupwindow",
                       "location=1,status=no,scrollbars=1,width=1200px,
                        height=768px,resizable=1,toolbar=0" )
                       mywindow.focus();
      }
}
```

Visual WebGui C# server code

```
LogonForm objLogonForm = new LogonForm(<…params…>);
objLogonForm.ShowDialog();
```

Quick review of the codes above:

- In the JavaScript sample, the URL points to a complete context detached URL.
- There is a great deal of work still to be done on client side and maybe even server side in order to send arguments to this dialog and handle them.
- Visual WebGui refers to a completely bound dialog that can accept any number or sort of parameters and can further interact with the other dialogs and controls using plain object oriented and server events.
- Security wise, sending parameters over using client scripting, automatically creates additional client vulnerabilities.

Visual WebGui provides you with the complete power to fully control your applications' logics, algorithms and data without using multilayered web development languages. All with a single layered object oriented, event driven code and a Form Designer (figure 6).

Gizmox Visual WebGui®
bringing AJAX & Silverlight to enterprises

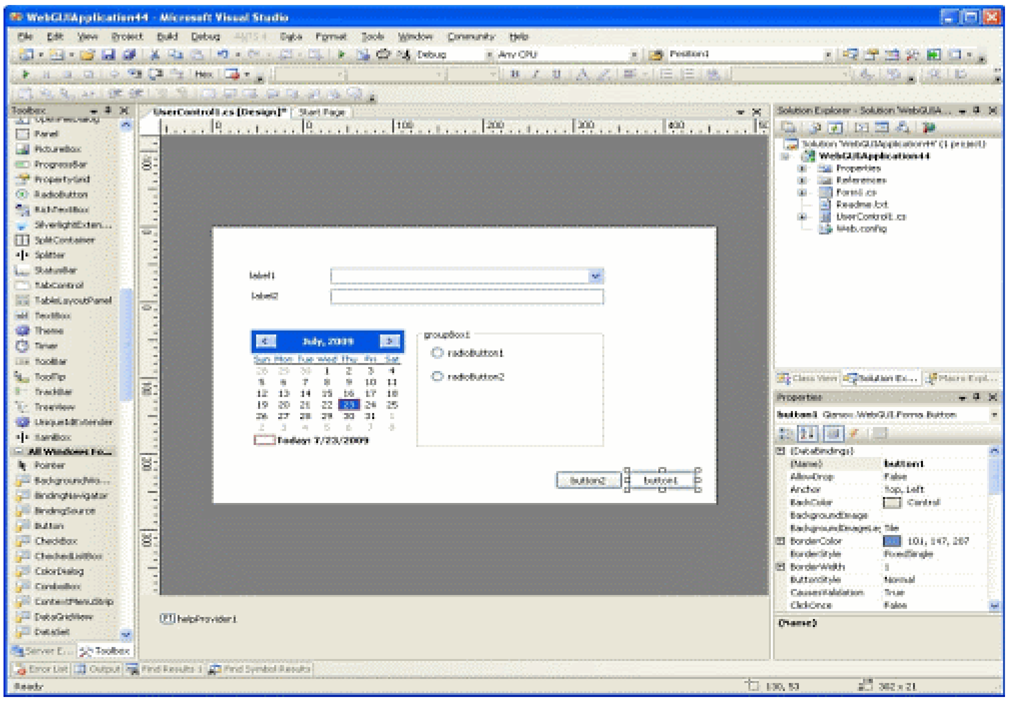

WebGUIApplication44 - Microsoft Visual Studio
Toolbox
UserControl1.cs [Design]*
Solution Explorer - Solution 'WebGUIA...
Properties
button1 Gizmox.WebGUI.Forms.Button
label1
label2
July, 2009
groupBox1
radioButton1
radioButton2
button2
button1
Today: 7/23/2009
helpProvider1
Ready

Fig. 6: WYSIWYG Forms designer

Graphic-designer-ready tools complement the development process with the ability to easily provide creative look & feel apps (figure 7).

- No CSS documents (although you can still control them)
- No HTML Hell (although you can still customize them)
- No Cross Browsers Incompatibilities

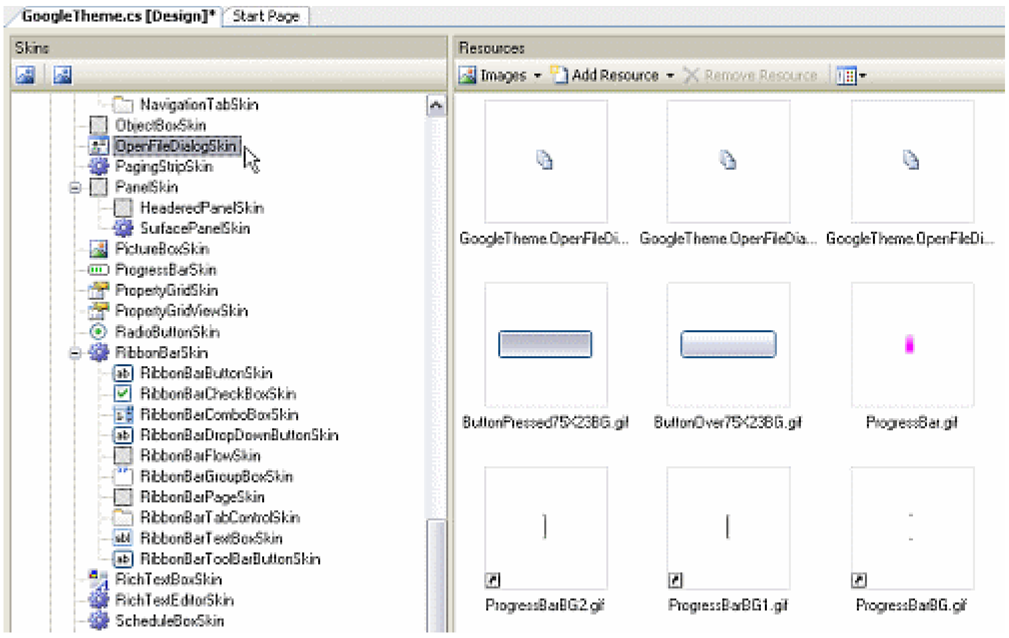


Fig. 7: Point & Click graphic-designer-ready tool for creative UIs

To be emphasized, although Visual WebGui abstracts away from client scripting and resources, you still have an organized access to those files and you can still override the entire scripting, CSS or HTMLs if needed. So being a web developer can help you provide even better applications (figure 8).

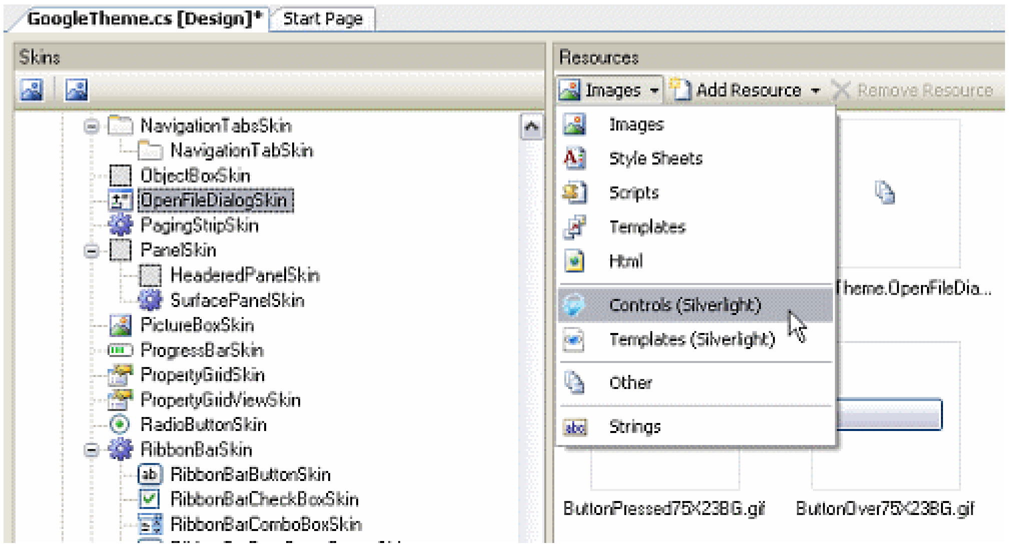


Fig. 8: Gain full control over client scripts and resources

API quality and convenience

Visual WebGui presents a well known API derived from desktop development APIs, however much extending it to the web.

ASP.NET including 3rd party components vendors offer verity of API and most of them are thorough, on the other hand there is no guideline or standard by which you always know how to interface with those controls and you can never know if you will be able to accomplish future tasks using common controls.

With Visual WebGui, not only you have a large set of controls, all fully featured and with AJAX native behavior, you can always easily extend those controls or customize them according to your needs.

Visual WebGui API is event driven; this means that your entire development will concentrate on handling events and performing the application logics.

For example, let's explore the simple task of handling a click event of a tree node located on the left hand of the application and filling up a list located on the right hand:

Standard Ajax approach:

- Locate the right way to handle the client event of clicking a tree-node.
- Make sure that you know how to read the required data from the selected tree-node.
- Option 1
  - Send the values over to the server using free style HttpXml or ASP.NET AJAX
- Option 2
  - Locate the way to affect the list control's mechanism and force it to perform server update call with parameters
- Grab the values on the server side and retrieve whatever data required to fill the list
- Send back the data or an inner HTML form to the client.
- Capture the callback on the client side with the data from the server
- Perform the local update of the list as needed

Visual WebGui new Ajax Approach:

- Attach the AfterSelect event of the tree view.
- Write event handler code to retrieve the data and at the same time fill it into the list using plain C# (as shown below).

```
private void treeView1_AfterSelect(object s,TreeViewEventArgs e)
{
  //Populate ListView
  while (objDataReader.Read())
  {
    for (int i = 0; i < objDataReader.FieldCount; i++)
    {
      ListViewItem objItem = new ListViewItem();
      objItem.Text = objDataReader.GetName(i).ToString();
      objItem.SubItems.Add(objDataReader[i].ToString());
      listView1.Items.Add(objItem);
    }
  }
}
```

* The entire action will be performed once on the server and then reflected back to the client at runtime.

Quick review of the codes above:

- Inefficiently long development process as opposed to a well defined events API usage.
- Imagine having more than one control to update as a result of the tree-node click event; then you would have to perform 2 requests/response roundtrips. It's easy to discover that Visual WebGui is also much more efficient at runtime since it goes back to the server once with no further client processing, then returns back once and updates the whole UI changes in this one roundtrip and with no post-backs.

Ability to use common and proven design patterns

Web architecture separates almost completely between the client and the server creating a decoupled architecture on which the developer should find ways to bridge and cover.

It is most common that well defined design patterns are much more complex to achieve on top of the traditional web architecture; for example:

- Data Validations: What kinds of validations should exist on the client side and what validations should be tested on the server? Maybe both?
- MVC: Where do we actually split between the client side's role and the server side role? In addition, in case we have got some logics on the client side, does this mean that we should also have parts of the controllers' mechanism on the client?(figure 9, 10)

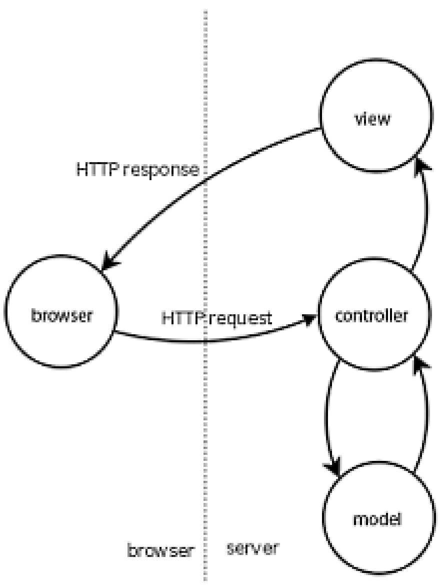


Fig. 9: Server based MVC

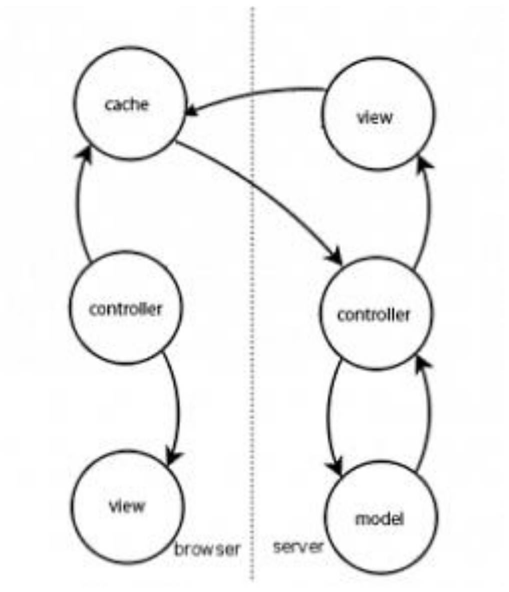


Fig. 10: Mixed client/server MVC

- Observer/Command: Which kinds of events should be raised to the server in order to bubble the events between the listeners?
- Service locator: Where should we keep the service locator code? On the client side on the server side? Maybe both?

A single layered development environment, were you write your entire code at a single coding language and handle it as an optimized event driven infrastructure on top of a complex, multilayered environment will make your life much easier in trying to apply quality design patterns.

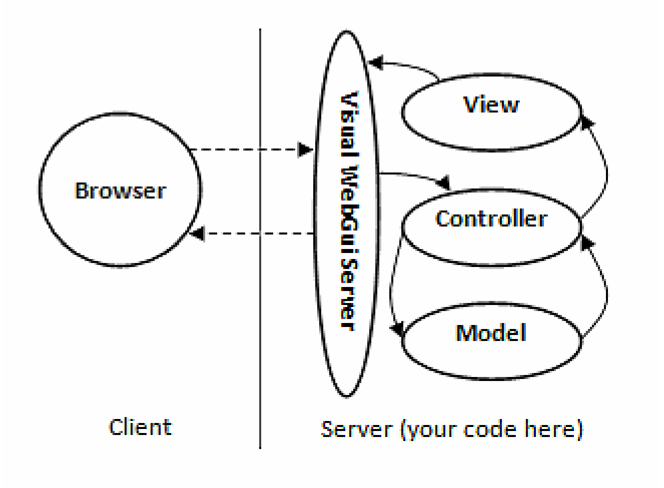
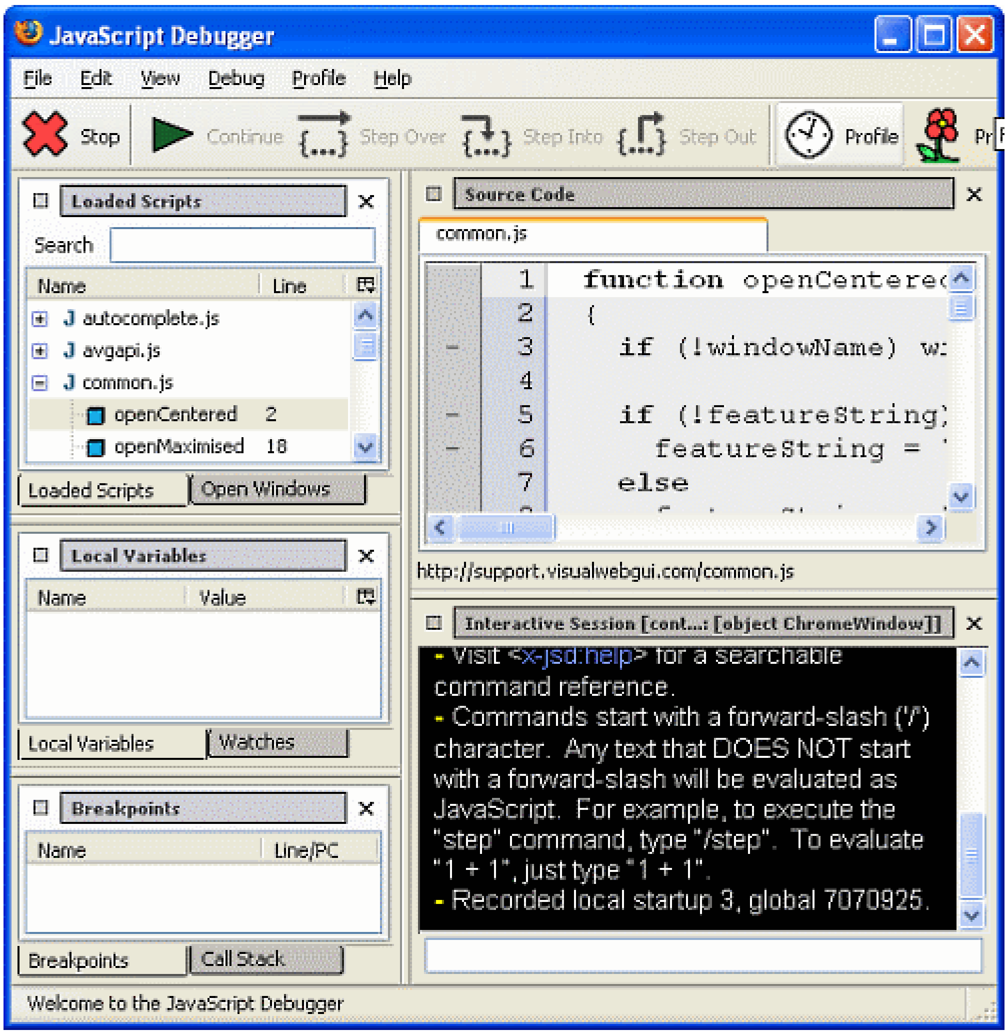


Fig. 11: Visual WebGui design pattern ready architecture

Debugging & testing efforts needed

Traditional web debugging requires:

- Server code debugging (using Visual Studio)
- Client-server communications debugging (figures 12-13 using Visual Studio and client development tools)
- Client scripting language debugging (figures 12-14 using client development tools)
- HTML and CSS actual rendering (figure 14 using client development tools)

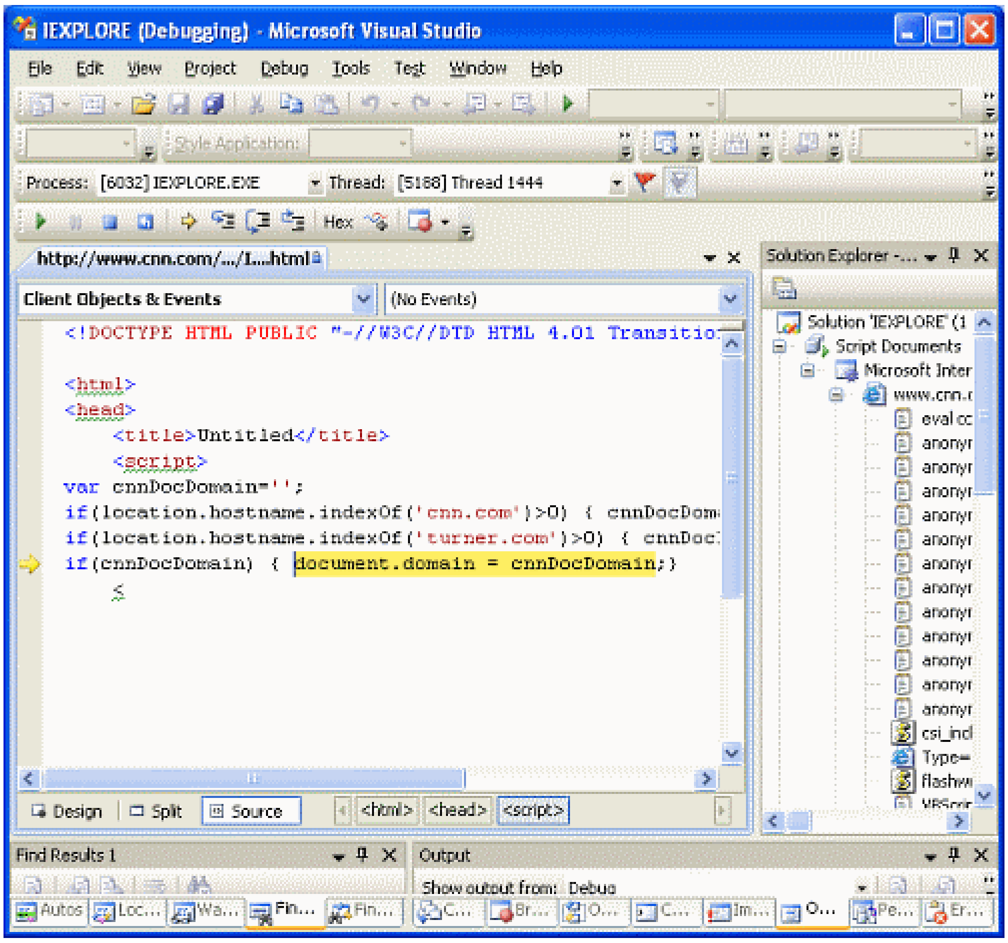


```
function openCentere
{
  if (!windowName) w:

  if (!featureString)
    featureString = '
  else
```

- Visit <x-jsd:help> for a searchable command reference.
- Commands start with a forward-slash ('/') character. Any text that DOES NOT start with a forward-slash will be evaluated as JavaScript. For example, to execute the "step" command, type "/step". To evaluate "1 + 1", just type "1 + 1".
- Recorded local startup 3, global 7070925.

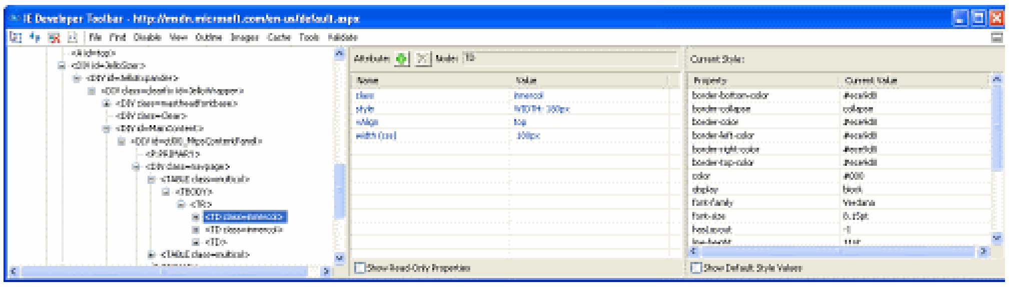
Fig. 12: Client JavaScript Debugger

Gizmox Visual WebGui® bringing AJAX & Silverlight to enterprises



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitio

<html>
<head>
    <title>Untitled</title>
    <script>
var cnnDocDomain='';
if(location.hostname.indexOf('cnn.com')>0) { cnnDocDom
if(location.hostname.indexOf('turner.com')>0) { cnnDoc
if(cnnDocDomain) { document.domain = cnnDocDomain;}
    <
```

Fig. 13: Visual Studio Script Debugger


IE Developer Toolbar - http://msdn.microsoft.com/en-us/default.aspx
File Find Disable View Outline Images Cache Tools Validate
Attribute:
Value:
Current Style:
Show Read-Only Properties
Show Default Style Values

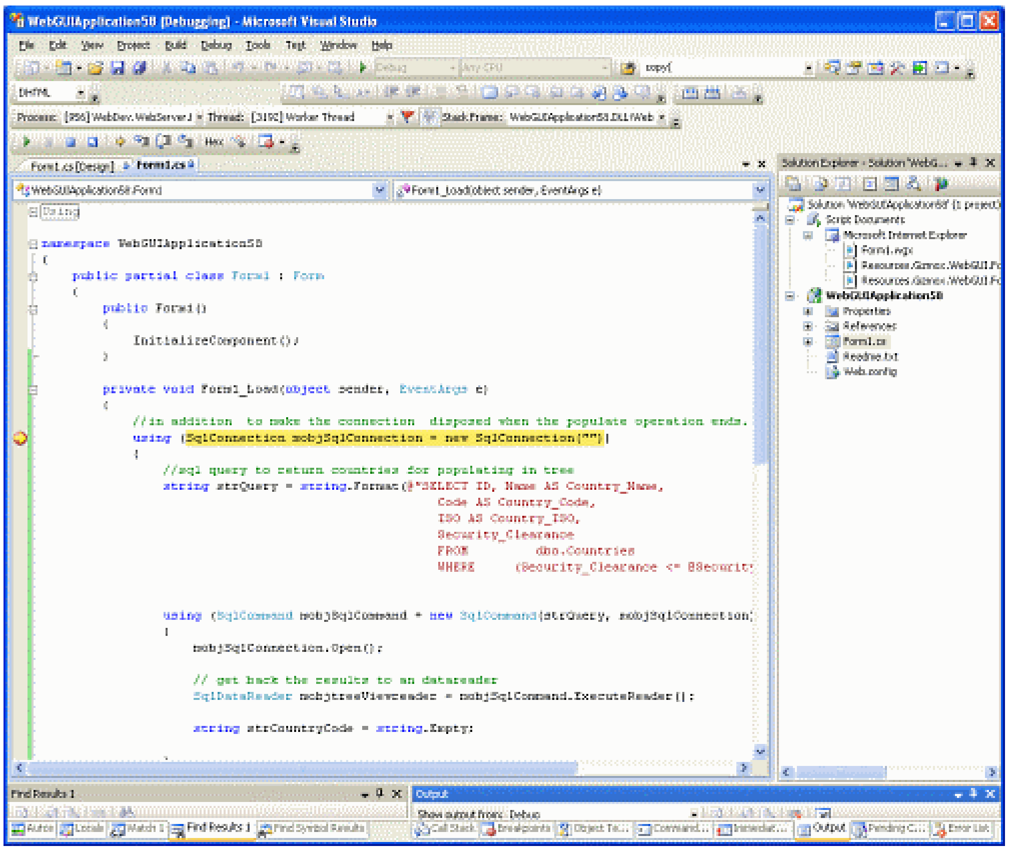
```
<A id=top>
<DIV id=JelloSizer>
  <DIV id=JelloExpander>
    <DIV class=clearfix id=JelloWrapper>
      <DIV class=mastheadFontbase>
      <DIV class=Clear>
      <DIV id=MainContent>
        <DIV id=ctl00_MtpsContentPanel>
          <!--PRIMARY-->
          <DIV class=navpage>
            <TABLE class=multicol>
              <TBODY>
                <TR>
                  <TD class=innercol>
                  <TD class=innercol>
                  <TD>
            <TABLE class=multicol>
```

Name	Value
class	innercol
style	WIDTH: 300px
vAlign	top
width (css)	300px

Property	Current Value
border-bottom-color	#ecefd8
border-collapse	collapse
border-color	#ecefd8
border-left-color	#ecefd8
border-right-color	#ecefd8
border-top-color	#ecefd8
color	#000
display	block
font-family	Verdana
font-size	0.75pt
hasLayout	-1
line-height	[illegible]

Fig. 14: HTML and CSS debuggers/inspectors (IE Dev-tool)

Visual WebGui debugging means debugging an object oriented C# code only (figure 15)


WebGUIApplication58 (Debugging) - Microsoft Visual Studio
File Edit View Project Build Debug Tools Test Window Help
Process: [956] WebDev.WebServer.I Thread: [3192] Worker Thread Stack Frame: WebGUIApplication58.DLL!Web
Form1.cs [Design] Form1.cs
WebGUIApplication58.Form1
Form1_Load(object sender, EventArgs e)
Solution Explorer - Solution 'WebG...
Solution 'WebGUIApplication58' (1 project)
Script Documents
Microsoft Internet Explorer
Form1.wgx
Resources.Gizmox.WebGUI.Fo
Resources.Gizmox.WebGUI.Fo
WebGUIApplication58
Properties
References
Form1.cs
Readme.txt
Web.config
Find Results 1
Output
Show output from: Debug
Autos Locals Watch 1 Find Results 1 Find Symbol Results
Call Stack Breakpoints Object Te... Command... Immediat... Output Pending C... Error List

```
Using

namespace WebGUIApplication58
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            //in addition  to make the connection  disposed when the populate operation ends.
            using (SqlConnection mobjSqlConnection = new SqlConnection(""))
            {
                //sql query to return countries for populating in tree
                string strQuery = string.Format(@"SELECT ID, Name AS Country_Name,
                                                  Code AS Country_Code,
                                                  ISO AS Country_ISO,
                                                  Security_Clearance
                                                  FROM         dbo.Countries
                                                  WHERE      (Security_Clearance <= @Securit

                using (SqlCommand mobjSqlCommand = new SqlCommand(strQuery, mobjSqlConnection
                {
                    mobjSqlConnection.Open();

                    // get back the results to an datareader
                    SqlDataReader mobjtreeViewreader = mobjSqlCommand.ExecuteReader();

                    string strCountryCode = string.Empty;
```

Fig. 15: Visual WebGui single layered C# code debugging

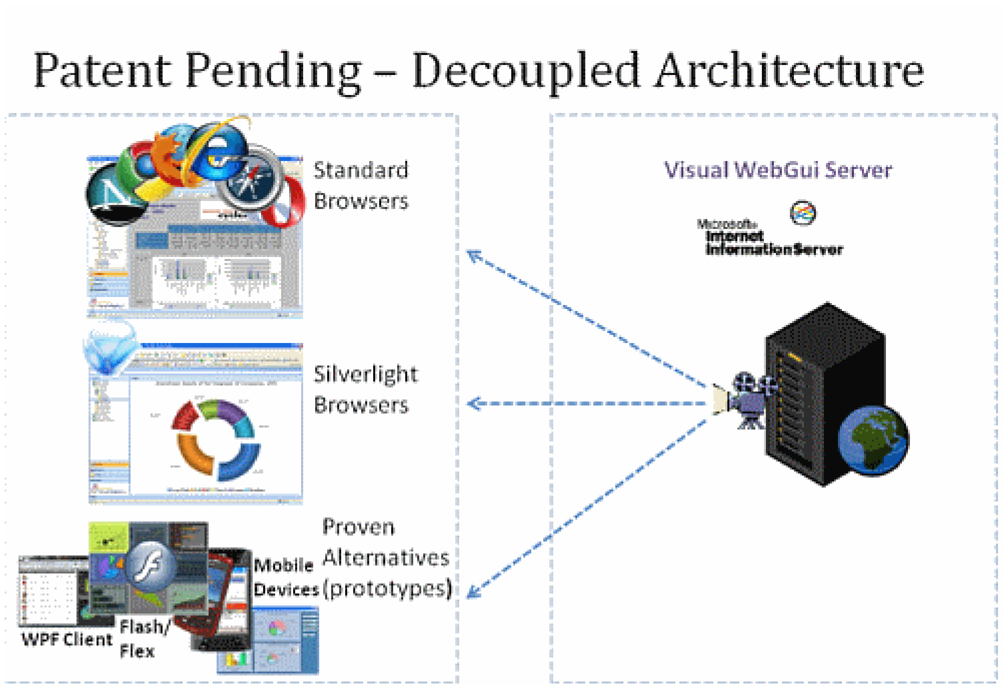
Technologies contemporariness

Assuming you are done with the development process, ending up with the work-of-art application on top of the traditional web development paradigms then your customers are pretty much locked into consuming the app. through DHTML browsers.

In addition, developing DHTML 4 and certain CSS standards means you will need to invest an additional large amount of time in order to upgrade your application to newer standards.

With Visual WebGui, you can use any .NET version (currently 2.0 or 3.5) on the server, while the client is completely agnostic. Visual WebGui provides alternatives for any standard DHTML clients with no installation supporting any browser as well as Silverlight enriched browsers with no additional development effort.

Visual WebGui will continue to maintain the contemporariness of both server and client technologies keeping you updated with the cutting edge technologies at all times and with minimal to non effort (figure 16).



Patent Pending – Decoupled Architecture
Standard Browsers
Silverlight Browsers
Proven Alternatives (prototypes)
Mobile Devices
WPF Client
Flash/ Flex
Visual WebGui Server
Microsoft Internet Information Server

Fig. 16: Cutting edge technologies on both client and server sides

Choosing an infrastructure

The following short explanation summarizes the major considerations of choosing an application development infrastructure that will ensure the customer's satisfaction and the quality of the product.

Infrastructure standards

In order to fit most of the customers' current infrastructures, we should make sure that we base our solution on a well known existing and proven infrastructures.

Visual WebGui is based on ASP.NET the deployment using XCOPY simple deployment.

Visual WebGui communication layer is based on plain XMLs over HTTP and supports any plain web browser Silverlight or any device with no specific installation!

Variety of data controls

In order to make sure that you are able to deliver any sort of UI needs and any common way possible way of presenting/updating and interacting within the application and more specifically its data, you will probably want to examine the out-of-the-box verity of data controls.

Visual WebGui contains a set of above 50 controls out-of-the-box and a set of complementing libraries with about 20 more additional controls, supplying you the complete suite for any common interaction and manipulate (figure 17).

In case you need more, you can either import in any ASP.NET based 3rd party control, create your own or customize one of the existing once (see Extensibility/Customizability & look & feel).

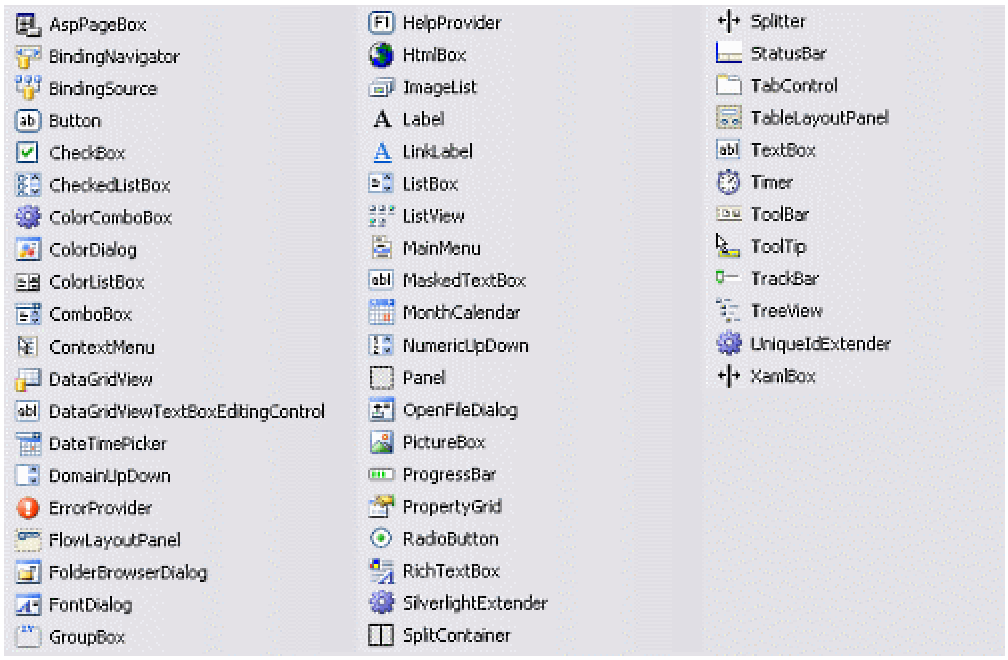
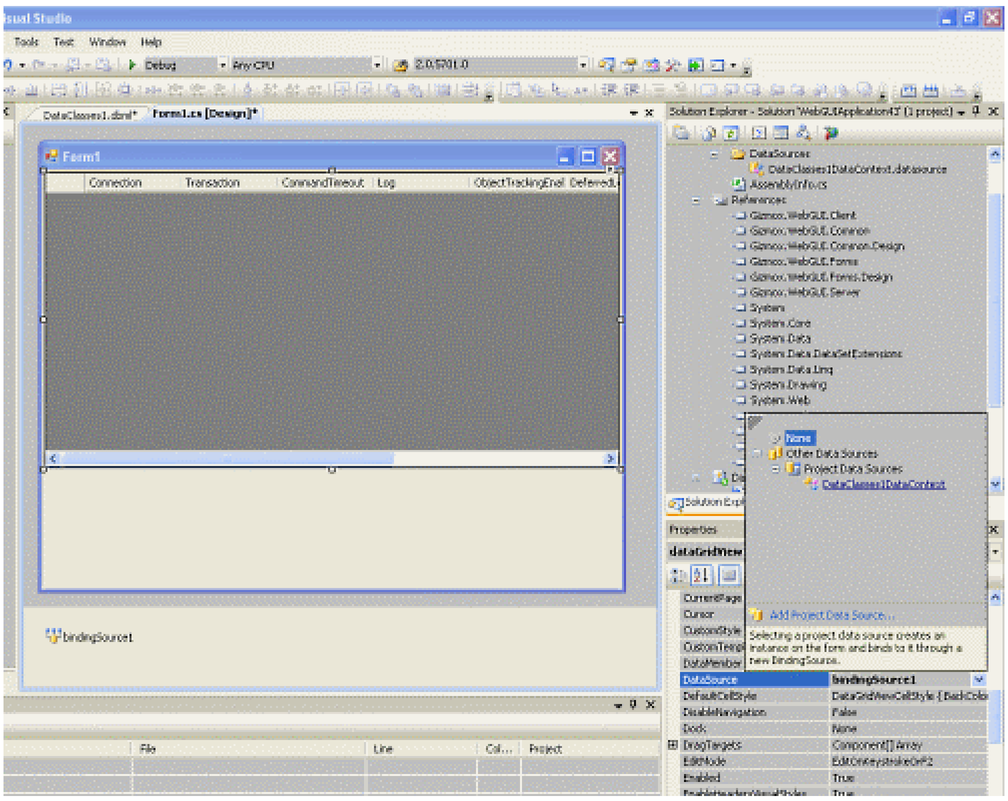

AspPageBox
BindingNavigator
BindingSource
Button
CheckBox
CheckedListBox
ColorComboBox
ColorDialog
ColorListBox
ComboBox
ContextMenu
DataGridView
DataGridViewTextBoxEditingControl
DateTimePicker
DomainUpDown
ErrorProvider
FlowLayoutPanel
FolderBrowserDialog
FontDialog
GroupBox
HelpProvider
HtmlBox
ImageList
Label
LinkLabel
ListBox
ListView
MainMenu
MaskedTextBox
MonthCalendar
NumericUpDown
Panel
OpenFileDialog
PictureBox
ProgressBar
PropertyGrid
RadioButton
RichTextBox
SilverlightExtender
SplitContainer
Splitter
StatusBar
TabControl
TableLayoutPanel
TextBox
Timer
ToolBar
ToolTip
TrackBar
TreeView
UniqueIdExtender
XamlBox

Fig. 17: Visual WebGui large set of controls out-of-the-box

Binding to data

Complete Data-binding options make it easy to concentrate on the business logics of the application as opposed to struggling with various techniques for binding your UI to data (figure 18).


Selecting a project data source creates an instance on the form and binds to it through a new BindingSource.

DataSource	bindingSource1
DefaultCellStyle	DataGridViewCellStyle { BackColo
DisableNavigation	False
Dock	None
DragTargets	Component[] Array
EditMode	EditOnKeystrokeOrF2
Enabled	True

Fig. 18: Complete data-binding experience

Extensibility/Customizability & look & feel

Visual WebGui's extensibility & customization options are served as productive driven tools to the developer. The mechanisms required to apply design changes and runtime abilities are wrapped into an optimized engine (figure 19).

Except for the WYSIWYG forms designer, Visual WebGui provides a visually simple solution to edit, re-create and customize Visual WebGui controls. In addition, an automatic tool for migration of 3rd party controls is also enabled immediately widening the available verity of controls with all the kinds of ASP.NET based controls by any of the existing providers.

Gizmox Visual WebGui®
bringing AJAX & Silverlight to enterprises

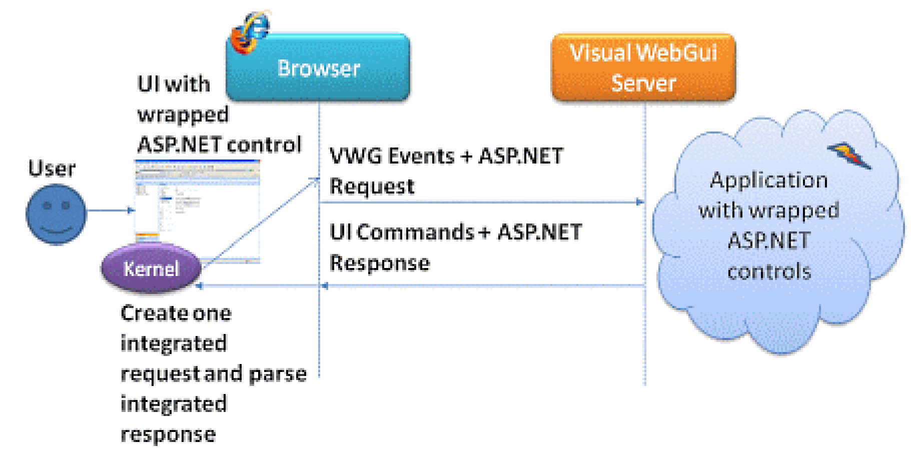

Browser
Visual WebGui Server
UI with wrapped ASP.NET control
User
Kernel
VWG Events + ASP.NET Request
UI Commands + ASP.NET Response
Application with wrapped ASP.NET controls
Create one integrated request and parse integrated response

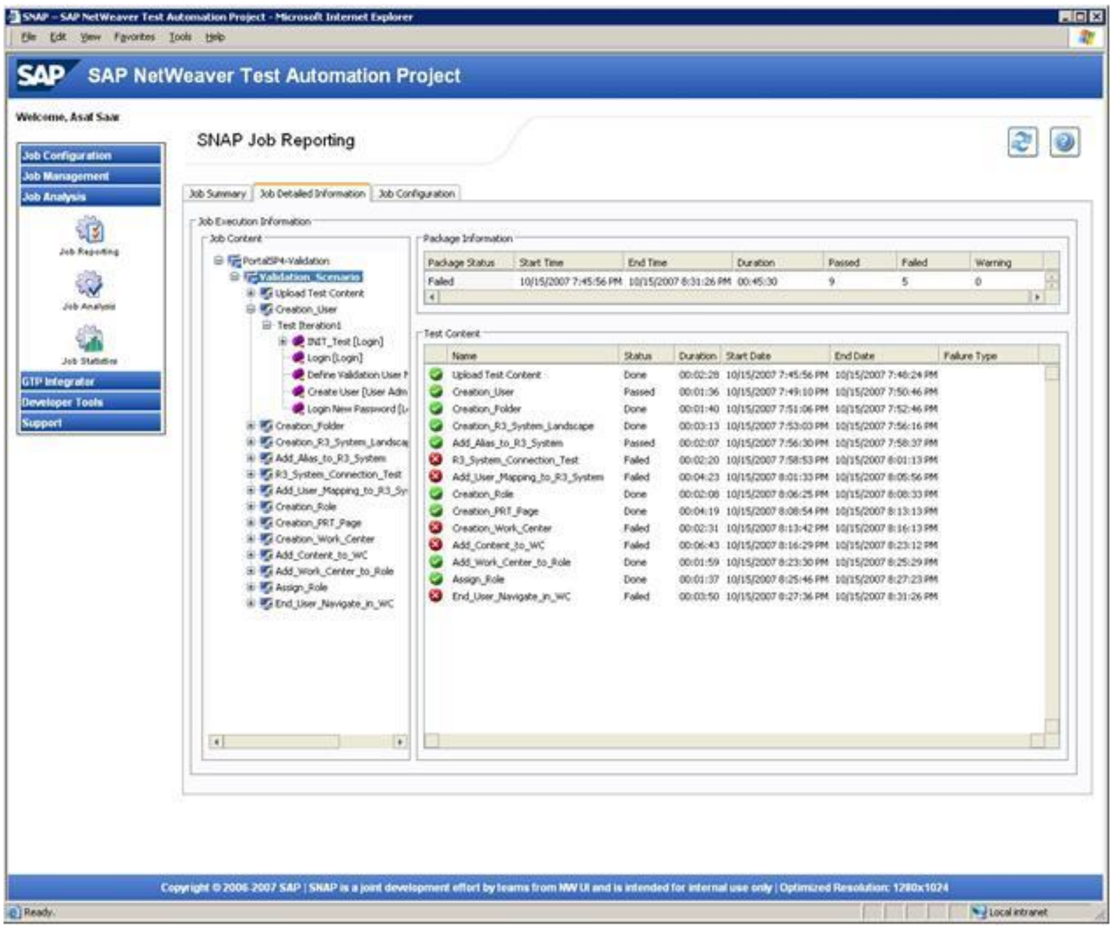
Fig. 19: Optimized ASP.NET controls interoperability

As for branding and UI customizations, you have the complete power to customize & brand the application using a point & click designer to change the look & feel completely according to the customers' requirements.

Integration/Interoperability

Based on a generic desktop compliant API and enjoying the fact that its ASP.NET based web under the hoods, Visual WebGui offers a very wide solution. Starting from stand alone applications development through mash-ups and ending-up in highly interactive and data centric add-ons.

Visual WebGui offers the ASP.NET FormBox control which enables ASP.NET based applications to contain Visual WebGui applications. Figure 20 show a large testing central application by SAP (called SNAP), which combines Visual WebGui with ASP.NET. The data centric and interactive part in the middle is Visual WebGui and the surrounding "frame" is ASP.NET.



Package Status	Start Time	End Time	Duration	Passed	Failed	Warning
Failed	10/15/2007 7:45:56 PM	10/15/2007 8:31:26 PM	00:45:30	9	5	0

	Name	Status	Duration	Start Date	End Date	Failure Type
✓	Upload Test Content	Done	00:02:28	10/15/2007 7:45:56 PM	10/15/2007 7:48:24 PM	
✓	Creation_User	Passed	00:01:36	10/15/2007 7:49:10 PM	10/15/2007 7:50:46 PM	
✓	Creation_Folder	Done	00:01:40	10/15/2007 7:51:06 PM	10/15/2007 7:52:46 PM	
✓	Creation_R3_System_Landscape	Done	00:03:13	10/15/2007 7:53:03 PM	10/15/2007 7:56:16 PM	
✓	Add_Alias_to_R3_System	Passed	00:02:07	10/15/2007 7:56:30 PM	10/15/2007 7:58:37 PM	
✗	R3_System_Connection_Test	Failed	00:02:20	10/15/2007 7:58:53 PM	10/15/2007 8:01:13 PM	
✗	Add_User_Mapping_to_R3_System	Failed	00:04:23	10/15/2007 8:01:33 PM	10/15/2007 8:05:56 PM	
✓	Creation_Role	Done	00:02:08	10/15/2007 8:06:25 PM	10/15/2007 8:08:33 PM	
✓	Creation_PRT_Page	Done	00:04:19	10/15/2007 8:08:54 PM	10/15/2007 8:13:13 PM	
✗	Creation_Work_Center	Failed	00:02:31	10/15/2007 8:13:42 PM	10/15/2007 8:16:13 PM	
✗	Add_Content_to_WC	Failed	00:06:43	10/15/2007 8:16:29 PM	10/15/2007 8:23:12 PM	
✓	Add_Work_Center_to_Role	Done	00:01:59	10/15/2007 8:23:30 PM	10/15/2007 8:25:29 PM	
✓	Assign_Role	Done	00:01:37	10/15/2007 8:25:46 PM	10/15/2007 8:27:23 PM	
✗	End_User_Navigate_in_WC	Failed	00:03:50	10/15/2007 8:27:36 PM	10/15/2007 8:31:26 PM	

Copyright © 2006-2007 SAP | SNAP is a joint development effort by teams from NW UI and is intended for internal use only | Optimized Resolution: 1280x1024

Fig. 20: An ASP.NET page which contains a Visual WebGui form

The opposite option is a Visual WebGui application being able to contain an ASP.NET application. This option is provided in the form of a control named AspPageBox in Visual WebGui (Figure 21).

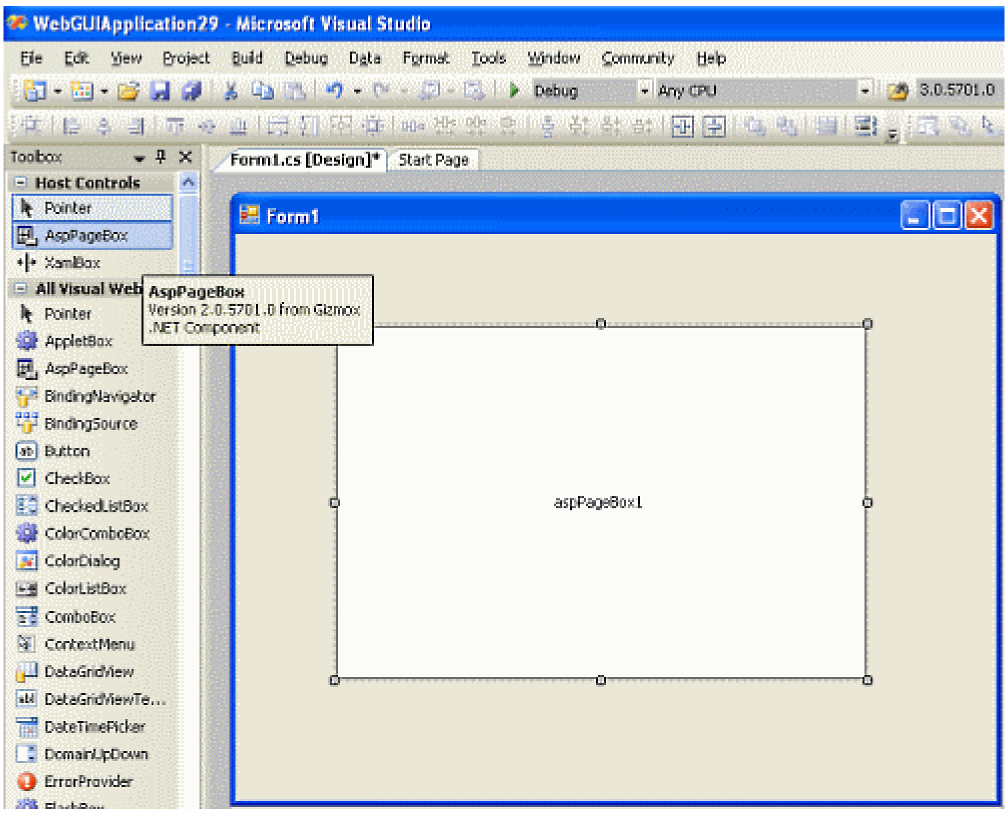


Fig. 21: A Visual WebGui form ability to contain an ASP.NET page

Performance considerations

Although Visual WebGui is a server centric architecture it optimizes the communication and the balance between client and server responsibilities and provides low latency and excellent performance. In highlights level, the following factors are producing this outcome:

- Low CPU usage on the server side low negotiation establishment time due to the existence of valid context at all times.
- Highly optimized communication protocol based on compressed deltas metadata and minimal commands.
- Leveraging the client machine power to minimize communication and throughput.

The effectiveness of the protocol is shown in the following graphs (figures 22-23) based on external testing performed by a performance specialist Microsoft MVP Mr. Wiktor Zychlah.

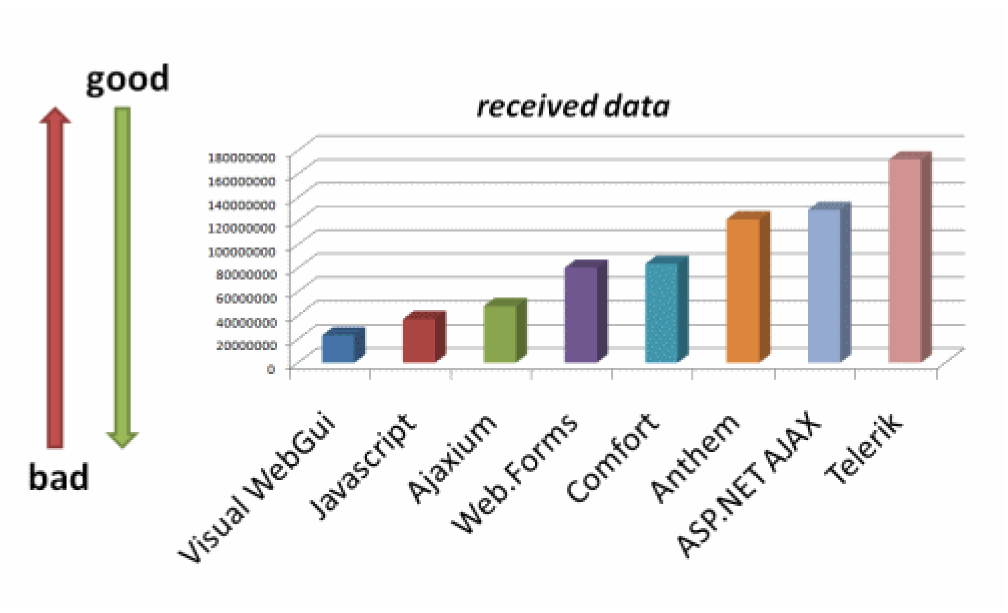


Fig. 22: Received sent data

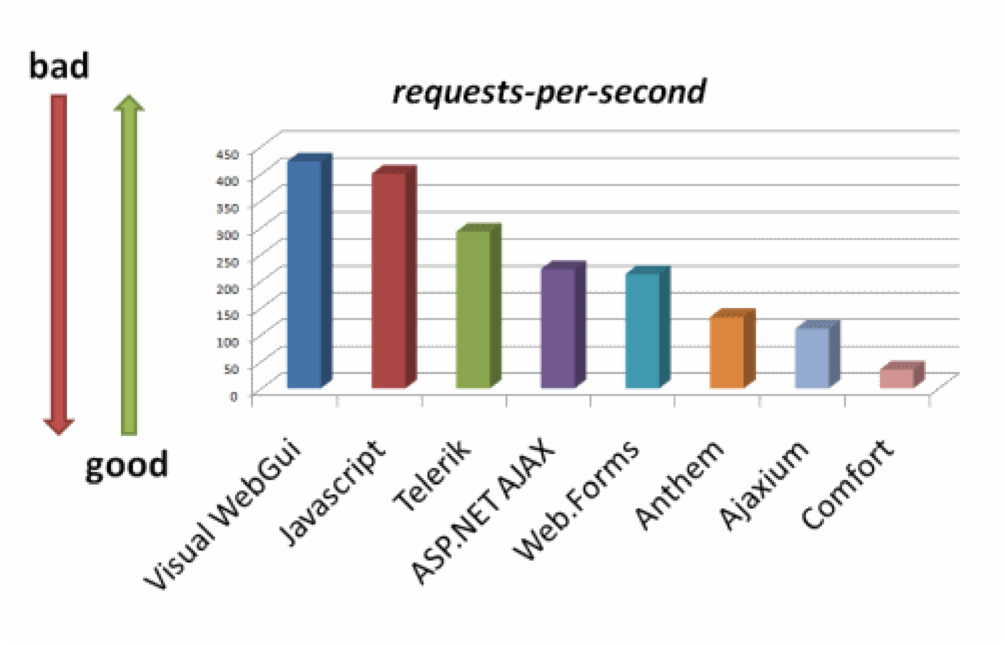


Fig. 23: Requests per second

Security factors

Visual WebGui Empty and static client with no ability to change logics is the preferred approach for enterprise business applications data security.

It is not that traditional web paradigms are not securable; however, the effort that is required to support such information safety level is significantly bigger.

In addition, encryptions/decryptions and obfuscations at runtime are costly and might affect performance so it is highly preferred avoiding them by-design.

Scalability

Due to the unique optimizations of the state memory and persistence model (memory serialization), Visual WebGui based solutions are fully scalable using a floating session and the average penalty for having a centralized persistent state is ~15ms per call. A schematic diagram is shown in figure 24. This fact makes Visual WebGui is an optimized solution for cloud computing architecture as well as on-premise or standard hosting deployments.

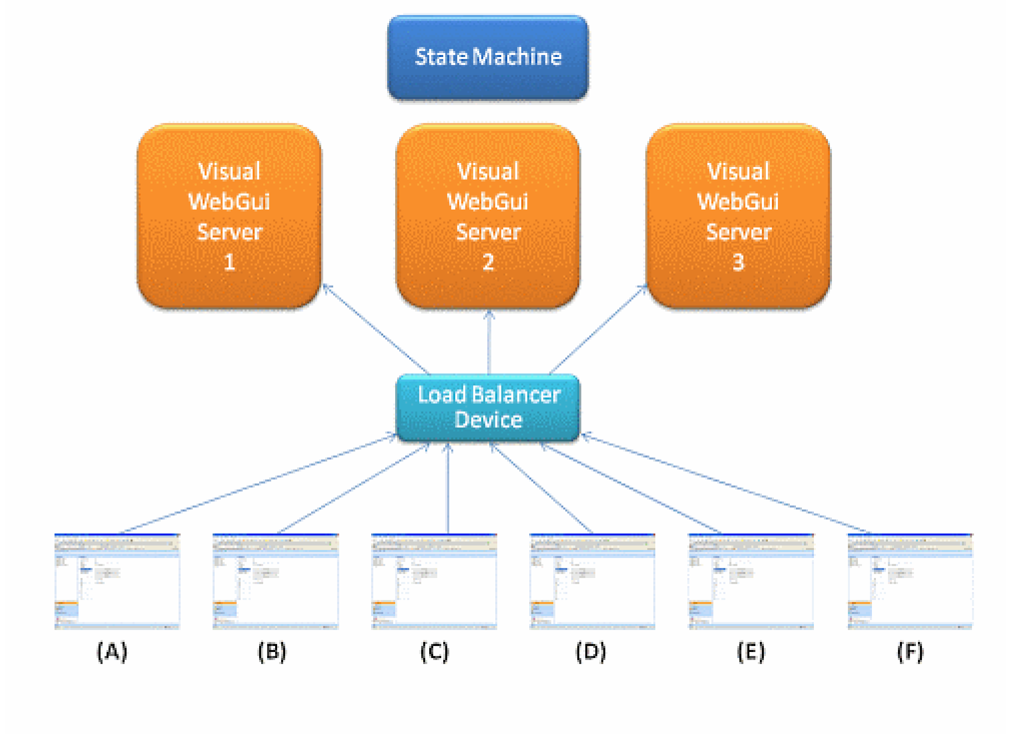


Fig. 24: Fully dynamic scalable and redundant solution

About the author:

Itzik Spitzen has over 13 years of experience in software development field of which lecturing, development and R&D management. Major profession is web development utilizing mainly Microsoft's technologies. Developed and managed a couple of very large scale software projects (such as BPM, ECM, Internet applications and frameworks). Holds an MSc degree in Computer Science and a Practical Engineering diploma.