# MapMatrix NetGIS WebControls Users Guide

## Version 2.3

# CONTENTS

# 1 Overview of MapMatrix NetGIS

MapMatrix NetGIS is a technology that has been designed to allow mapping and GIS information to be quickly and easily integrated into software applications. Any forms or browser based, internet, intranet or local application can utilize the MapMatrix NetGIS technology. NetGIS products sets the new standard for reducing total cost of creation, ownership and project lifecycle span for web based GIS enabled applications.

## 1.1 MapMatrix NetGIS Features

MapMatrix NetGIS is provides GIS information by distributing data from MapServers via Web Services to applications that have been GIS enabled utilizing the NetGIS application objects. Many GIS features are provided including:

- Display map data from a variety of data formats.
- Add custom data formats with the NetGIS Custom Data Adapters.
- Very fast display capabilities with many window and pan commands.
- Display imagery from a variety of data formats including JPEG, MrSid, Tiff, etc.
- Perform Geocoding.
- Create thematic displays.
- Allow the selection of GIS objects and retrieve tabular information for the graphical feature.
- Join disconnected databases. This allows for remote clients to apply database information that is stored separate from the GIS data.
- Perform buffer zone analysis.
- View GIS data from multiple NetGIS MapServers. This allows the application to view information from any NetGIS MapServer and view it as if it were from one source.
- Integrate GIS into existing applications without requiring a total rewrite of the application.

MapMatrix NetGIS is a very powerful system that provides the distribution of GIS applications and data in a fast and flexible fashion.

## 1.2 Operating System Requirements

Before you install the MapMatrix NetGIS Application Objects or WebControls your machine must meet the following requirements.

- You must be running a version of Microsoft Windows compatible with the .NET Framework and ASP.NET. At the time of this writing you can use:

  - Windows NT 4.0 (with Service Pack 4 or greater installed)
  - Windows 2000 (all versions supported)
  - Windows XP (all versions supported)

- You must have Microsoft IIS (Internet Information Server) installed.

- You must have the Microsoft .NET Framework SDK installed

  Skip this step if you have Visual Studio.NET installed.
  If not you can visit the Microsoft .NET Framework SDK download page. At the time of this writing the .NET Framework SDK is available for download at the URL below.

  **http://www.microsoft.com/downloads/details.aspx?FamilyID=9b3a2ca6-3647-4070-9f41-a333c6b9181d&DisplayLang=en**

- To run the mock Service Request sample application you must have MDAC 2.6 or greater installed. At the time of this writing the latest version of MDAC can be downloaded from the URL below.

  **http://www.microsoft.com/data/**

## 1.3 The MapMatrix Architecture

The MapMatrix NetGIS Application Development System is based upon the latest in multi-tier application development architecture. MapMatrix applications are built on a true client/server architecture. The MapMatrix MapServer creates and delivers maps to client applications. The client applications are built with the MapMatrix NetGIS application products. Communication to and from the MapServer and the Client applications is performed through the MapMatrix NetGIS Web Services. Each of these are described below.

### 1.3.1 NetGIS MapServer

The MapServer receives and processes requests from the GIS Web Service (made upon request by SDK method calls). It is responsible for drawing maps and processing queries for attribute data. All GIS information is generated from a NetGIS MapServer. The server can be accessed through a local area network or via the Internet.

### 1.3.2 NetGIS MapServer Custom Data Adapters

The NetGIS MapServer reads GIS data from external sources by utilizing a specialized technique called GIS Data Late Binding. The implementation of this technique is accomplished with NetGIS Custom Data Adapters. Custom Data Adapters are software libraries that are built to read data from different data formats. If it is possible to read data through any vendor API or with custom programming, the data format can be supported and distributed with the NetGIS MapServer. These can be custom built by anyone and attached to a MapServer. This allows anyone to build a Custom Data Adapter for any format and use this GIS data with a NetGIS MapServer.

### 1.3.3 NetGIS Web Services

The NetGIS Web Service acts as a broker between the NetGIS MapServer and the NetGIS application objects. Web Services are easily made available across joined and disjoined networks with low risk to security and reduced maintenance requirements. The NetGIS WebService is an XML web service that provides the means for software to connect with other software applications. Any application that supports Web protocols such as HTTP, XML and SOAP can communicate with the NetGIS WebService and apply information created from a NetGIS MapServer.

### 1.3.4 NetGIS Application Objects

The NetGIS Application Objects supply software developers with a set of tools that provide a simple method for interacting with a NetGIS MapServer. The

NetGIS Objects provides this capability with an intuitive structure and command set. Communication from the NetGIS Objects with a MapServer is via the NetGIS Web Services. All GIS commands and queries and processed through this method.

### 1.3.5 NetGIS WebControls

The NetGIS WebControls builds upon the NetGIS Application Objects to provide an ASP.NET and Visual Studio.NET friendly set of components for web programmers. These controls simply many of the tasks that Web developers must face when developing applications such as state management and event handling. The NetGIS WebControls also simplify the management of retrieving GIS information from multiple MapServers.

Use of the NetGIS WebControls require a license to the NetGIS Application Objects (except for the trial version).

# 2 Installing and Configuring NetGIS Products

## 2.1 Licensing the NetGIS Products

The trial versions of the software do not require a license. For all runtime versions of the MapMatrix NetGIS products are licensed separately, for more information on these and other products are licensed please visit our website at http://www.GatewayHorizons.com or call use at 281-312-5600.

## 2.2 Obtaining a License Key

The trial version of the software does **not** require a license key. The trial version will only work with the Gateway Horizons GIS Web Services (it only connects to the NetGIS WebServices at www.GatewayHorizons.com). If a license is purchased the software can communicate with any MapMatrix NetGIS Web Service.

Before you install the runtime versions of the MapMatrix NetGIS software you must have a valid license key. If you are evaluating MapMatrix on trial basis you will need to go to the Gateway Horizons website at http://www.GatewayHorizons.com and fill out a new user registration form.

## 2.3 Installing the NetGIS Software

The NetGIS Application Objects and WebControls make use of MSI (Microsoft Installer Services). As part of this service installed packages will be audited by Windows and can be removed using Add/Remove programs from Control Panel. When you run NetGIS Setup on screen instructions will explain each step. If you experience problems with install feel free to contact Gateway Horizons support for assistance.

Upon a successful install, you will have a new program group appear on your “All Programs” menu called “NetGIS Application Objects” or “NetGIS WebControls”. Inside of these groups you will find shortcuts to this documentation, license manager control panel and sample applications.

# 3 The NetGIS WebControls

The MapMatrix NetGIS WebControls are ASPNet Web Controls that have been built to facilitate the implementation of the MapMatrix NetGIS Application Objects inside of WebForm applications.

The MapMatrix NetGIS WebControls provides Web developers with a set of controls and objects that can plug into Visual Studio.NET, WebMatrix or any .NET capable visual IDE. Through built in session management and rich featured drag and drop controls, NetGIS WebControls sets the new standard for reducing total cost of creation, ownership and project lifecycle span for web based GIS enabled applications.

WebControls are utilized in conjunction with the MapMatrix NetGIS Application Objects and builds upon a powerful set of objects that will simplify GIS application development. With these products accessing data in a MapMatrix MapServer, any application can become GIS capable.

## 3.1 How to Use the NetGIS WebControls

The NetGIS WebControls make use of the NetGIS Application Objects. In order to fully understand how applications are developed see the NetGIS Application Objects documentation.

The WebControls provide a simple capability for creating Web Browswer based applications with the Microsoft .Net Framework. These controls have been designed to create GIS enabled Web applications with a minumum of effort.

### 3.1.1 The WebControls

There are two primary controls that are used to create GIS enabled applications, the MapSession control and the MapImage control. Both controls must be used to create a functioning GIS application.

The MapSession control establishes a connection with a NetGIS MapServer and through which all commands are issued to the MapServer. It contains a NetGIS Command object (see the NetGIS Application Objects documentation).

The MapImage control provides the location for the map to be displayed and handles user interaction with the map, such as selecting elements and identifying locations and areas for windowing. The MapImage can be be bound to one or more MapSessions, therefore can communicate with one or more MapServers simultaneously.

### 3.1.2 Placing WebControls

The WebControls are placed on a ASPNet form by drag and drop or by coding the html portion of the ASPNet Web Form. For detailed information refer to the .Net Framework documentation on ASPNet Web Form Programming. You can find information on this on the Miscrosoft MSDN web site at:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnservice/html/service02202002.asp

Important Items regarding placing controls:

- More than one MapImage control and MapSession can be placed on a Web Form.
- Multiple MapSessions can be associated with a single MapImage.
- Configuration information must be supplied to the MapSession control. This information is stored in an XML file and specifies which MapServer the MapSession control communicates with.
- The MapServer connection is created when the objects are instantiated, that is, when the application first starts communication is establish per the information in the configuration file.
- Synchronization of multiple MapSessions is performed automatically by the MapImage control when commands are issued via the MapImage.

### 3.1.3 Steps in creating a GIS Enabled Application

When utilizing the NetGIS WebControls the following steps should be followed:

1. Place a MapSession and MapImage control on the ASPNet Web Form.
2. Set the property for the MapSession Control indicating which configuration file to use. This identifies what MapServer will be communicated with.
3. Add code to the Form_Load event binding the MapImage to the MapSession.
4. Add code to the Form_Load event inside of a not PostBack condition block adding layers to the MapSession (its Connection) and do an initial refresh of the MapImage
5. Add controls and code as desired to support any other GIS functions, such as Window, Pan, Identify, etc.

More detailed information on the objects is contained in the following sections.

## 3.2 MapSession Control

The MapSession Control is the control that creates a connection to a NetGIS WebService (MapServer) and provides the ASPNet Web Application with the means for communicating with the MapServer.

The MapSession acts as a wrapper control around the NetGIS Command and Connection objects to provide an ASP.NET friendly programming interface. MapSession controls are bound to a MapImage WebControl to complete the configuration necessary for performing GIS functions. Many MapSession objects an be bound to a single MapImage control, this allows a single MapImage control to show results from multiple MapServers.

### 3.2.1 Placing the MapSession Control on a WebForm

If you are using the .Net IDE then simply drag and drop the control onto the Web Form. If you are using another editor or development environment you can place the control on the page by including the assembly at the top of the html portion of the aspx page.

```
<%@ Register TagPrefix="cc1" Namespace="MapMatrix.ASPNetControls"
Assembly="MapMatrix.ASPNetControls" %>
```

Then place a reference to the object at the appropriate location in the body of the page.

```
<cc1:mapsession id="MapSession1" style="Z-INDEX: 102;
      LEFT: 288px; POSITION: absolute; TOP: 160px"
      runat="server">
</cc1:mapsession>
```

The physical location of the control is not important as there is no graphical representation of the MapSession during run time.

### 3.2.2 Configuring the MapSession Control

The only property that must be set on the MapSession control is the **`ConfigFile`** property. This must be set to the name of the XML File that holds the configuration information the specifies what MapServer will be accessed. The XML File must reside in the ASPNet project directory. If more than one MapSession controls will be used in the application, give each control a different name (this assumes that there are some differences between the different MapServer connections. Here is the MapMatrix.config file that connects to the Gateway Horizons NetGIS WebServices.

```
<?xml version="1.0" standalone="yes" ?>
<mapmatrixconfig>

      <mapservermanagers>

            <machine url="www.GatewayHorizons.com" id="mapserver1" />
            <service name="MapMatrixWS"/>
```

```
            <customer id="MunicipalExample" />

      </mapservermanagers>

      <log>
            <error file="log.txt" path="c:\code\log\" />
      </log>
</mapmatrixconfig>
```

This config file should be added to your project and should have a .config extension. This file must exist in order for the MapSession object to work properly. The content of the file is as follows:

- The `machine` tag identifies the url that the NetGIS WebServices exist on.
- The `service` tag identifies the name of the NetGIS WebService that will be used for the Connection.
- The `customer` tag indicates what configuration will be loaded from the NetGIS Web Service.

These three tags are placed within the `<mapservermanagers>` block and are mandatory. A valid connection to the Web Service will not be made unless this information is provided.

The `<log>` block provides information that can be used for debugging. The path information is for the local machine (the machine that hosts the ASPNet application). This information is not required.

### 3.2.3 Properties and Methods

The MapSession exposes its NetGIS Connection and Command objects as public properties. For many MapServer commands you will rely on the MapMatrix NetGIS Command object. For example, if you wanted to add a new layer the code would look like this:

#### *3.2.3.1.1 C#*

```
if (!(Page.IsPostBack))
{
   // Add the layers to the map session
   MapSession1.Command.Layers.Add(MapSession1.Command.GetAvailableLayer("Roads"));
```

#### *3.2.3.1.2 VB*

```
If (Not Page.IsPostBack) Then

   ' Add the layers to the map session
   MapSession1.Command.Layers.Add(MapSession1.Command.GetAvailableLayer("Roads"))
```

For complete details on how to use the MapControls Command and Connection objects see the MapMatrix NetGIS Objects documentation.

**Property Summary**

| | |
|---|---|
| ActiveCommand | The active command. This is from the enumeration ActiveCommandTypes |
| ApplicationID | A key that is persisted with the WebForm. This is used to ensure that the state of the client application stays in sync with the MapServer sessions. |
| Command | The NetGIS Command object. The Command object is how commands are submitted to a MapServer. See the NetGIS Objects documentation for more information. |
| ConfigFile | The name of the XML configuration file (.config) that contains the MapServer connection information. |
| Connection | The NetGIS Connection object. This Connection object contains the actual connection information to the MapServer. |
| Enabled | Indicates whether or not a connection should be established for this MapSession object. This can be useful for debugging applications that contain more than one MapSession. |
| Height | The initial height (in pixels) that will be used for the Map. This is an optional property as the size of the map image can be set several ways. |
| Width | The initial width (in pixels) that will be used for the Map. This is an optional property as the size of the map image can be set several ways. |

**Method Summary**

| | |
|---|---|
| GetActiveLayer | Returns the Active Layer. |
| RaiseUpdateMapEvent | Causes the MapImage to be refreshed with the latest image returned from the MapServer. |
| SetActiveCommand | Sets the active command with an enumerated ActiveCommandType |
| SetActiveLayer | Sets the active layer. This indicates which layer operations such as identify and find operate on. |

## 3.3 MapImage WebControl

The NetGIS MapImage Control binds to a MapSession Control and responds to update events as needed. The MapImage has built in functionality to respond to click events. For example, if the ActiveCommand is ZoomIn then a zoom operation will be performed when the MapImage is clicked; if the ActiveCommand is ElementInformation then an Identify command is performed on the ActiveLayer.

In addition to the click event handling, the MapImage manages multiple MapSession objects and can also provide dynamic image resizing and zoom to window (the user can drag a rectangle on the map image area and the zoom operation will zoom to that region).

### 3.3.1 Placing the MapImage Control on a WebForm

If you are using the .Net IDE then simply drag and drop the control onto the Web Form. If you are using another editor or development environment you can place the control on the page by including the assembly at the top of the html portion of the aspx page.

```
<%@ Register TagPrefix="cc1" Namespace="MapMatrix.ASPNetControls"
Assembly="MapMatrix.ASPNetControls" %>
```

Then place a reference to the object at the appropriate location in the body of the page.

```
<cc1:mapimage id="MapImage1" style="Z-INDEX: 101; LEFT: 88px;
      POSITION: absolute; TOP: 160px"
      runat="server" Height="280px" Width="350px"></cc1:mapimage>
```

The size of the image as defined here will not be represented in design mode but will be represented properly when the application starts.

### 3.3.2 Configuring the MapImage Control

The MapImage control contains several properties that need to be set. The width and height property can be set to any pixel size that you desire. This sets the size that the MapImage is sized to at run time.

It is not important to set other properties in the designer unless multiple MapSession Controls are being used on this Web Form. If this is the case, the properties `HttpImageDirectory` and `TemporaryImageDirectory` must be set. These two properties provide the MapImage control with information that is

needed to perform the merging of the images returned from the multiple Map Servers and embed them for viewing in the current Web Form.

### 3.3.3 Binding to a MapSession

The MapImage control must be aware of which MapSession objects it uses for communication to the MapServer(s). This is accomplished by binding the MapSession object to the MapImage object. This code is placed inside of the Page_Load event.

#### *3.3.3.1.1 C#*

```
Private void Page_Load(object sender, System.EventArgs e)
{
      MapImage1.MapSessionCollection.Add(MapSession1);
      MapImage1.Bind();
```

#### *3.3.3.1.2 VB*

```
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

      MapImage1.MapSessionCollection.Add(MapSession1)
      MapImage1.Bind()
```

The MapSessionCollection can contain more than one reference to a MapSession object. This allows data from multiple map sessions to be displayed on a single map image.

### 3.3.4 Properties, Methods and Events

The MapImage control provides certain properties, methods, and events that can simplify the interaction with a MapServer, particularly if multiple MapServers (MapSessions) are being used. Where the property or method duplicates a MapSession or Command property or method, it can be used in their place. In other words when a MapImage property or command is set or used it is propagated down to the appropriate MapSession or Command.

**Property Summary**

| | |
|---|---|
| ActiveCommand | The active command. This is from the enumeration ActiveCommandTypes. This can be used in place of the MapSession ActiveCommand. |
| Height | The height in pixels that the MapImage will be at run time. |
| HighlightOn | Determines whether or not elements are displayed |

| | |
|---|---|
| | with a Highlight when selected. |
| HttpImageDirectory | The local path used for the image file name reference when building the http for client viewing. This must be a valid url on the local server. |
| ImageURL | The URL of the current map image as returned by the MapServer. |
| TemporaryImageDirectory | A valid path on the local server that is used performing image operations. This is used to perform the overlay of the images from multiple MapSessions |
| Width | The height in pixels that the MapImage will be at run time. |

**Method Summary**

| | |
|---|---|
| Bind | Binds a MapSession to the MapImage |
| PanByDirection | Performs a Pan and updates the Image. |
| Refresh | Refreshes the image with the latest image from the MapServer |
| ResizeMap | Changes the current size of the MapImage. This can be used in place of the Command object command. |
| SynchronizeSessions | Synchronizes the ground area between all MapSessions. This can be used if one session has had a command performed and the display area has changed and it is necessary to get all of the other MapSessions display area to be the same. |

**Event Summary**

| | |
|---|---|
| Click | Raised when the MapImage has been clicked. Any user code is executed after MapServer commands have been processed. |

# 4 Walkthrough – Building an ASP.NET Address Lookup Application

This section walks you though the steps of building an Address lookup application in ASP.NET, you will find a similar sample application, with complete source code in the \samples subdirectory where you installed WebControls or on our web site.

PLEASE NOTE: This walkthrough makes reference to building the Address Lookup app with Visual Studio.NET. Note that Visual Studio.NET is not required to build ASP.NET applications with the .NET Framework. If you do not have Visual Studio.NET you can download the .NET Framework SDK free from Microsoft's msdn website.

Visit the Gateway Horizons website http://www.GatewayHorizons.com developer zone for additional sample applications.

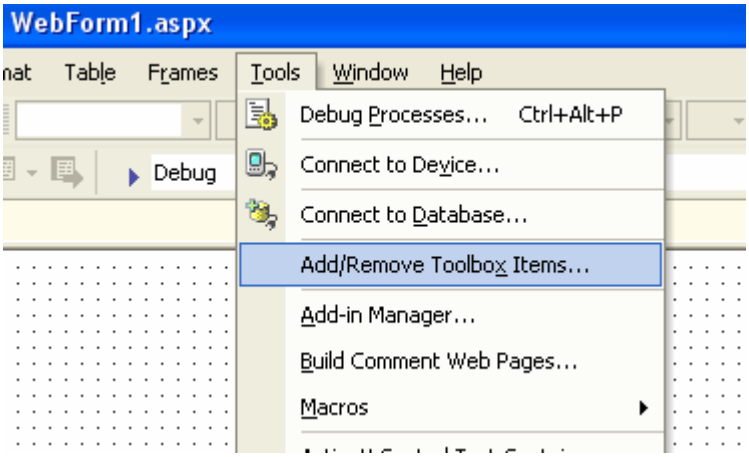
## Step One – Create a new ASPNet Project

The first step is to create a new ASPNet  project.

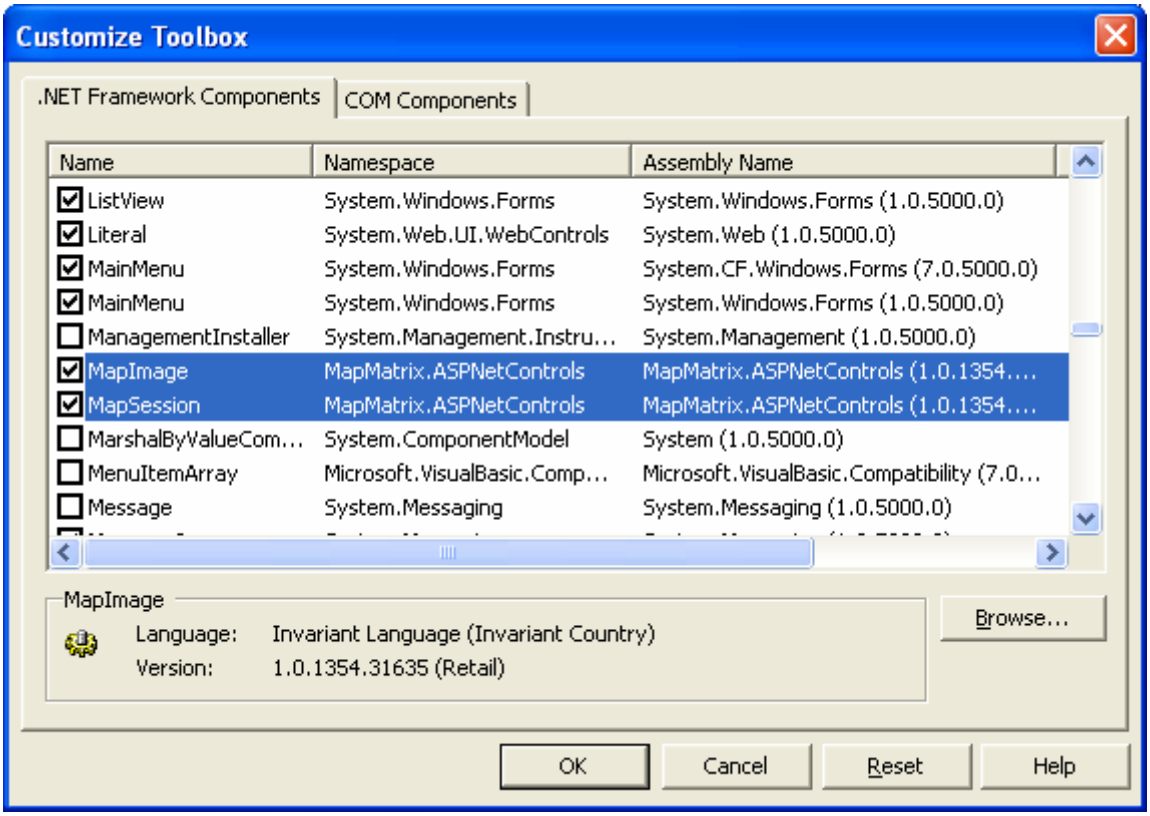
Open Visual Studio .NET and create a new ASP.NET WebForms application project. This can be done as either a Visual Basic or C# project.

## 4.1 Step Two – Add the NetGIS Controls to the Toolbox

If the MapSession and MapImage controls do not appear in the toolbox you must add them. From the Tools menu select Add Remove Toolbox Items.



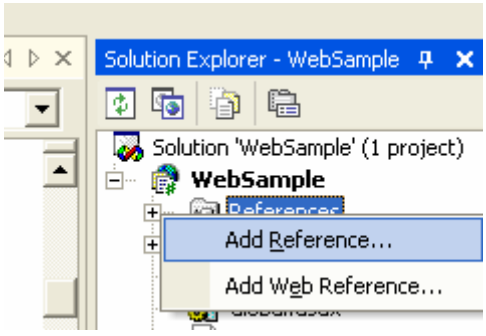

Browse to the directory that contains the MapMatrix NetGIS libraries and select the `MapMatrix.ASPNetControls.dll` library. The MapImage and MapSession controls will now appear in the Customize Toolbox dialog.



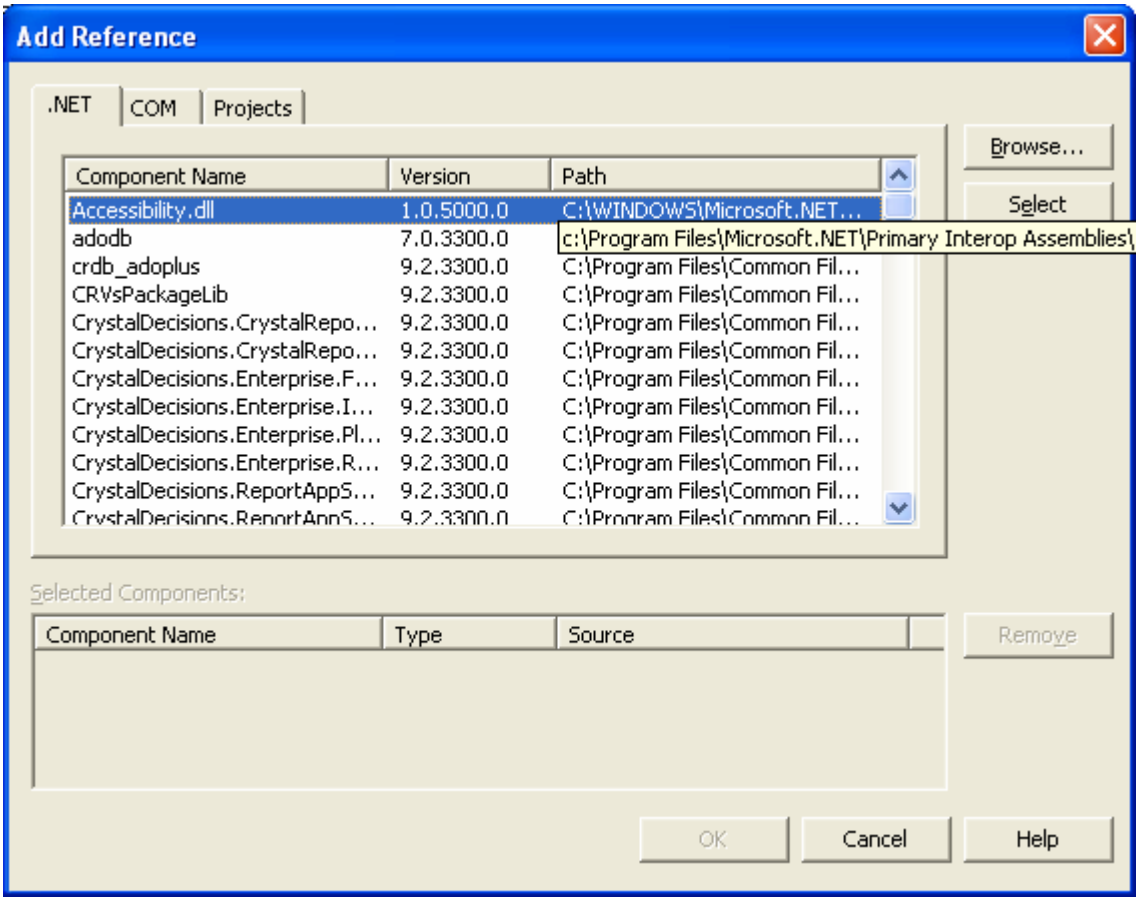

Click the OK button and the controls will be added to the ToolBox. The tools will appear at the bottom of the WebForms section on the ToolBox menu.

## 4.2 Step Three – Add references to the NetGIS Objects

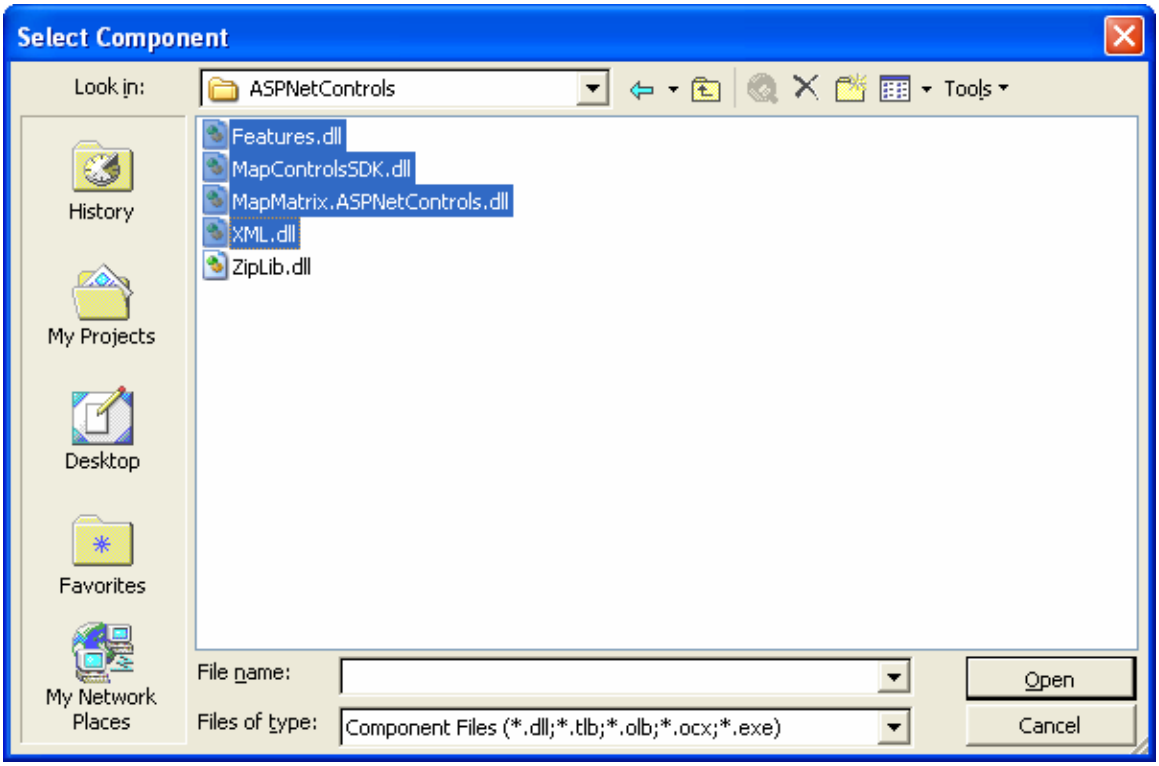
The ASPNet project must have a reference the NetGIS objects. Add them by right-clicking on the References folder in the solution explorer.



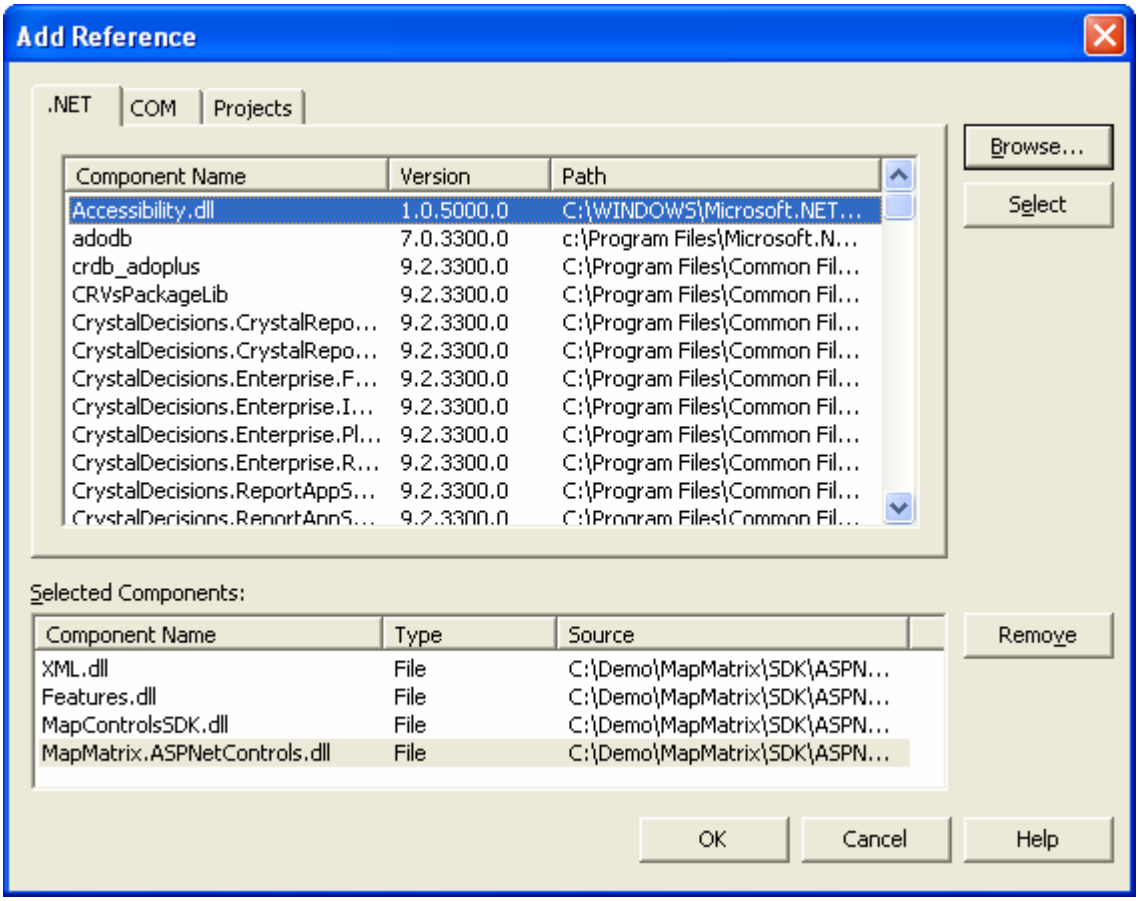

Select Add Reference from the pop-up menu. The following menu will appear.



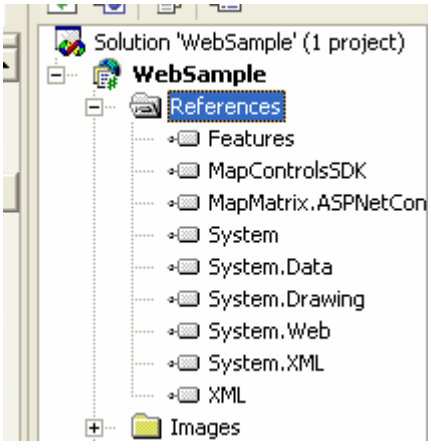

Click the Browse button and navigate to the location the contains the MapMatrix NetGIS Libraries. Select the `Features.dll`, `MapControlsSDK`.dll, `MapMatrix.ASPNETControls.dll` and `XML.dll` files.




Click open then OK on the following dialog.




This will add the references to the Solution Explorer. Expand the References folder and you should see the files.

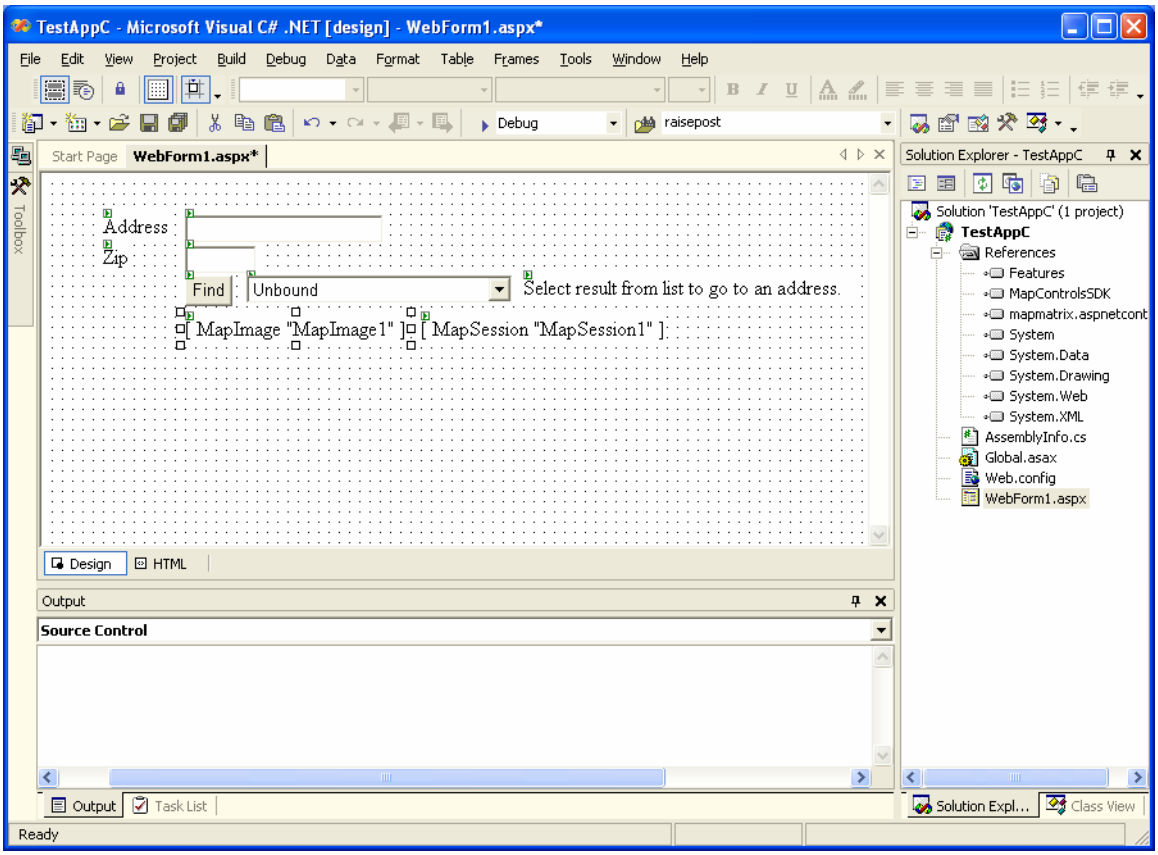



You are now ready to develop the application.

## 4.3 Step Four – Add Controls to the Web Form

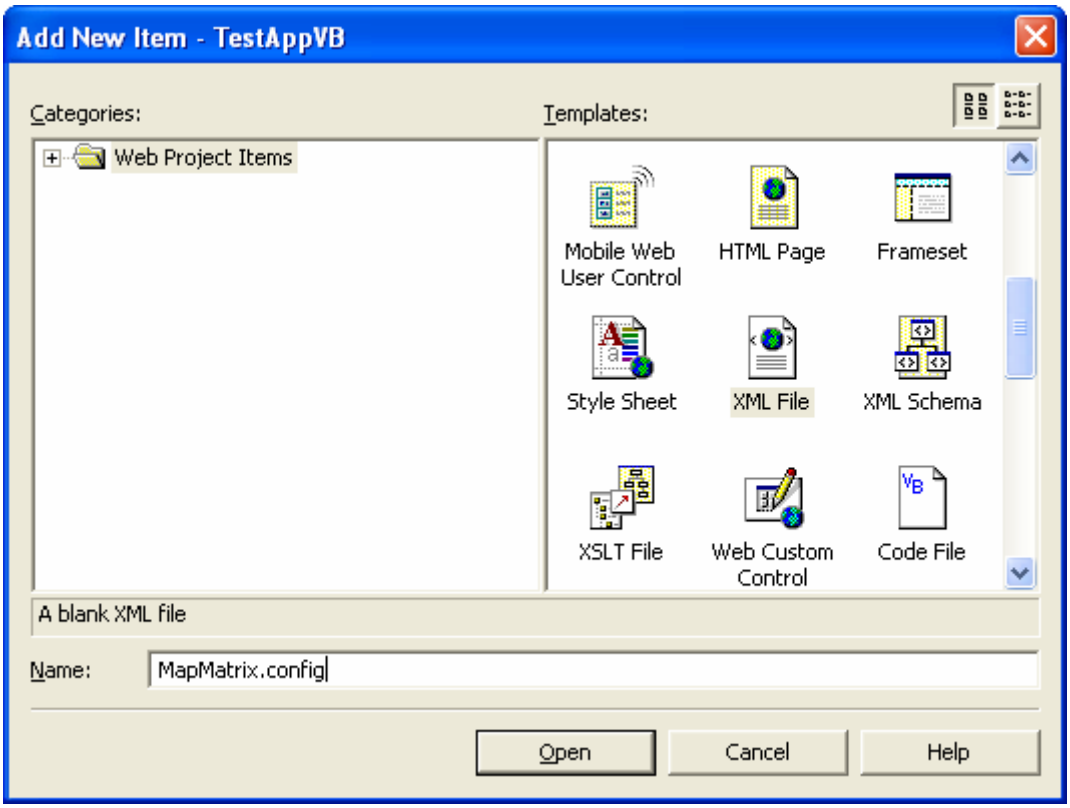
Using the form designer, setup your form like the one pictured below.

1. Drag a WebControl Label on the WebForm. Set the **Id** to "lblAddress" and its **Text** to "Address".
2. Drag another label and set **Id** to "lblZip" and its **Text** to "Zip".
3. Place another label with **Id** = "lblStatus" and its **Text** = "Select result from list to go to an address."
4. Place a WebControl TextBox next to lblAddress and name it txtAddress.
5. Place another TextBox next to lblZip and name it txtZip.
6. Place a Button below txtZip and name it btnFind and make its **Text** = "Find".
7. Place a DropDownList next to the Find button and name it ddAddresses. Size it so it can contain a large string (this will contain potential matches). Set the AutoPostback Property to true.
8. Place a MapImage below the Find button (its name will default to MapImage1). Set its Width = 350 and Height = 280.
9. Place a MapSession next to the MapImage (its name will default to MapSession1). The ConfigFile property will default to MapMatrix.config.



## 4.4 Step Five – Creating the MapMatrix.config

The MapMatrix.config file must be created with the appropriate connection information to the Gateway Horizons NetGIS Web Services. Add this file to the project by right-clicking on the Project in the solution explorer and selecting Add New Item.




Select the XML File from the Templates and name it MapMatrix.config. Add the following content.

```
<?xml version="1.0" standalone="yes" ?>
<mapmatrixconfig>

	<mapservermanagers>

		<machine url="www.GatewayHorizons.com" id="mapserver1" />
		<service name="MapMatrixWS"/>
		<customer id="MunicipalExample" />

	</mapservermanagers>

	<log>
		<error file="log.txt" path="c:\code\log\" />
	</log>
</mapmatrixconfig>
```

Save the file and close it. This information lets the MapSession object know what server it will be talking to.

## 4.5 Step Six – Adding References to the Code

Open the code view for the Web Form. You can do this by clicking one of the code icons (main toolbar or above solution explorer) or by right clicking the form and selecting View Code or by double clicking the page.

At the class level add the appropriate references to the MapMatrix class libraries.

#### *4.5.1.1.1 C#*

```
using MapMatrix.MapControls;
using MapMatrix.Features;
```

#### *4.5.1.1.2 VB*

```
Imports MapMatrix.MapControls
Imports MapMatrix.Features
```

## 4.6 Step Seven – Bind the MapSession to the MapImage

Navigate to the Page_Load event and add the code to perform the Bind. This code makes the MapImage aware of what MapSessions (what Map Servers) it is using.

#### *4.6.1.1.1 C#*

```
private void Page_Load(object sender, System.EventArgs e)
{
      MapImage1.MapSessionCollection.Add(MapSession1);
      MapImage1.Bind();
}
```

#### *4.6.1.1.2 VB*

```
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    MapImage1.MapSessionCollection.Add(MapSession1)
    MapImage1.Bind()
End Sub
```

This is the only code requirement that the MapMatrix NetGIS controls have at run time. If this code is not executed each time a page is loaded, the controls will not work properly.

## 4.7 Step Eight – Initialize the Map

When an application is first started several things must be done to establish what data the application works with. In order for map information to display, layers must be loaded to the current session. When the initial connection is made to the

Map Server a list of layers that are on the Map Server are placed in the AvailableLayers Collection. A layer is added only the first time that the page is loaded (it is persisted for the duration of the session in the control). Because of this you should only add the layer if the page is not being loaded from a post back. The modified Page_Load event is shown below.

#### *4.7.1.1.1 C#*

```
private void Page_Load(object sender, System.EventArgs e)
{
   MapImage1.MapSessionCollection.Add(MapSession1);
   MapImage1.Bind();

   if (!(Page.IsPostBack))
   {
      // Get a reference to the Command object
      MapMatrix.MapControls.Command cmd = MapSession1.Command;

      // Set the size in pixels of the map image
      cmd.ResizeMap(350,280);

      cmd.Layers.Add(cmd.GetAvailableLayer("Aerial Photography"));

      cmd.Layers.Add(cmd.GetAvailableLayer("Parcels"));
      cmd.Layers.Add(cmd.GetAvailableLayer("Roads"));

      // Set default to highlight elements that have been identified
      MapImage1.HighlightOn = true;

      // Issue the command to the map server
      MapSession1.UpdateMap();

      // Bind the Map to the MapImage
      MapSession1.RaiseUpdateMapEvent();
   }
}
```

#### *4.7.1.1.2 VB*

```
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

   MapImage1.MapSessionCollection.Add(MapSession1)
   MapImage1.Bind()

   If (Not Page.IsPostBack) Then

      ' Get a reference to the Command object
      Dim cmd As MapMatrix.MapControls.Command = MapSession1.Command

      ' Set the size in pixels of the map image
      cmd.ResizeMap(350, 280)

      cmd.Layers.Add(cmd.GetAvailableLayer("Aerial Photography"))
      cmd.Layers.Add(cmd.GetAvailableLayer("Parcels"))
```

```
        cmd.Layers.Add(cmd.GetAvailableLayer("Roads"))

        ' Set default to highlight elements that have been identified
        MapImage1.HighlightOn = True

        ' Issue the command to the map server
        MapSession1.UpdateMap()

        ' Bind the Map to the MapImage
        MapSession1.RaiseUpdateMapEvent()

    End If

End Sub
```

This code makes a reference to the Command object in the MapSession and adds layers to the Layers collection. If a Layer is in the Layers collection it can be displayed or manipulated with other GIS commands. A single command is issued to send the requests to the Map Server and the MapImage is refreshed. The application can now be run and you should see a map.

The default ActiveCommand for the MapImage object is ZoomIn. This means that if you click on the MapImage a ZoomIn is sent to the server and you will see a new image. Try it.

## 4.8 Step Nine – Add the Address Lookup to the Find button

The address lookup will be performed when the Find button is used. To perform this task we must add code to the btnFind click event. The simplest way to get to the code section for the click event is to open the WebForm in design mode and double click the find button. This will open the code page with the following code.

#### *4.8.1.1.1 C#*

```
private void btnFind_Click(object sender, System.EventArgs e)
{

}
```

#### *4.8.1.1.2 VB*

```
Private Sub btnFind_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnFind.Click

End Sub
```

In this method the GeoCode request (address lookup) will be submitted to the MapServer, the results will be persisted the WebForms viewstate and the dropdown will be populated with the results. The modified code is shown below.

#### *4.8.1.1.3 C#*

```
private void btnFind_Click(object sender, System.EventArgs e)
{
      // query for geocode results
      MapMatrix.MapControls.GeoCodeResultList results = null;

      try
      {
            results = MapSession1.Command.GetGeoCode(
                  new MapMatrix.MapControls.GeoCode(
                        txtAddress.Text, txtZip.Text, 40, 5, 15)
                        );
      }
      catch (Exception exc)
      {
            lblStatus.Text = exc.Message;
      }

      // Persist any Geocode results to viewstate
      ViewState["__GeocodeResults"] = null;

      // if result set isn't empty, pull up a map and display map
related UI controls
      if (results != null)
      {
            if (results.Count > 0)
            {
                  // bind results to combobox
                  ddAddresses.DataSource =

      MapMatrix.MapControls.Utilities.ConvertToMSDataTable(results);

                  ViewState["__GeocodeResults"] =
MapMatrix.MapControls.Utilities.ConvertToMSDataTable(results);

                  ddAddresses.DataTextField = "Address";
                  ddAddresses.DataValueField = "Address";
                  ddAddresses.DataBind();

      ZoomToGeoCode(MapMatrix.MapControls.Utilities.ConvertToMSDataTabl
e(results));
            }
            else
            {
                  lblStatus.Text = "Could not find a match.";
            }
      }
}
```

#### *4.8.1.1.4 VB*

```
Private Sub btnFind_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnFind.Click

   ' query for geocode results
   Dim results As MapMatrix.MapControls.GeoCodeResultList = Nothing

   Try
     results = MapSession1.Command.GetGeoCode(New
MapMatrix.MapControls.GeoCode(txtAddress.Text, txtZip.Text, 40, 5, 15))

   Catch exc As Exception

     lblStatus.Text = exc.Message
   End Try

   ' Persist any Geocode results to viewstate
   ViewState("__GeocodeResults") = Nothing

   ' if result set isnt empty
   ' pull up a map and display map related UI controls
   If Not results Is Nothing Then

      If results.Count > 0 Then

         ' bind results to combobox
         ddAddresses.DataSource =
MapMatrix.MapControls.Utilities.ConvertToMSDataTable(results)

         ViewState("__GeocodeResults") =
MapMatrix.MapControls.Utilities.ConvertToMSDataTable(results)

         ddAddresses.DataTextField = "Address"
         ddAddresses.DataValueField = "Address"
         ddAddresses.DataBind()


ZoomToGeoCode(MapMatrix.MapControls.Utilities.ConvertToMSDataTable(resu
lts))

      Else
         lblStatus.Text = "Could not find a match."
      End If
   End If

End Sub
```

This code does refer to another method that needs to be created. Just below this method code the following.

#### *4.8.1.1.5 C#*

```
private void ZoomToGeoCode(System.Data.DataTable table)
{
   if (table != null)
```

```
    {
        DataRow currentRow = table.Rows[ddAddresses.SelectedIndex];
        double groundX = Convert.ToDouble(currentRow["CoordinateX"]);
        double groundY = Convert.ToDouble(currentRow["CoordinateY"]);
        int windowSize = 500;

        MapSession1.Command.ZoomToGround(groundX, groundY, windowSize);
        MapSession1.RaiseUpdateMapEvent();
    }
}
```

#### *4.8.1.1.6 VB*

```
Private Sub ZoomToGeoCode(ByVal table As System.Data.DataTable)

  If Not table Is Nothing Then

    Dim currentRow As DataRow = table.Rows(ddAddresses.SelectedIndex)
    Dim groundX As Double = Convert.ToDouble(currentRow("CoordinateX"))
    Dim groundY As Double = Convert.ToDouble(currentRow("CoordinateY"))
    Dim windowSize As Integer = 500

    MapSession1.Command.ZoomToGround(groundX, groundY, windowSize)
    MapSession1.RaiseUpdateMapEvent()
  End If
End Sub
```

This method performs an additional call to the MapServer to change the display window to be 500 feet around the center of the address that has been located. This application will now run. Key in an address and click the Find button. Try an address such as “600 Wild Rose” or “400 Honeysuckle”. When the map displays you will see the names of other roads that you can try.

## 4.9 Step Ten – Change window to selected address

The code in the previous step populates the drop down control with a set of possible candidates that match the address that was keyed in. To relocate the window to another address in the drop down code must be entered to the SelectIndexChanged event. To code in this event double click the drop down from the designer. The following code will appear.

#### *4.9.1.1.1 C#*

```
private void ddAddresses_SelectedIndexChanged(object sender,
System.EventArgs e)
{

}
```

#### *4.9.1.1.2 VB*

```
Private Sub ddAddresses_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ddAddresses.SelectedIndexChanged

End Sub
```

When the application detects a _SelectedIndexChanged event it will perform a ZoomToGeoCode method call. This was coded in the previous step.

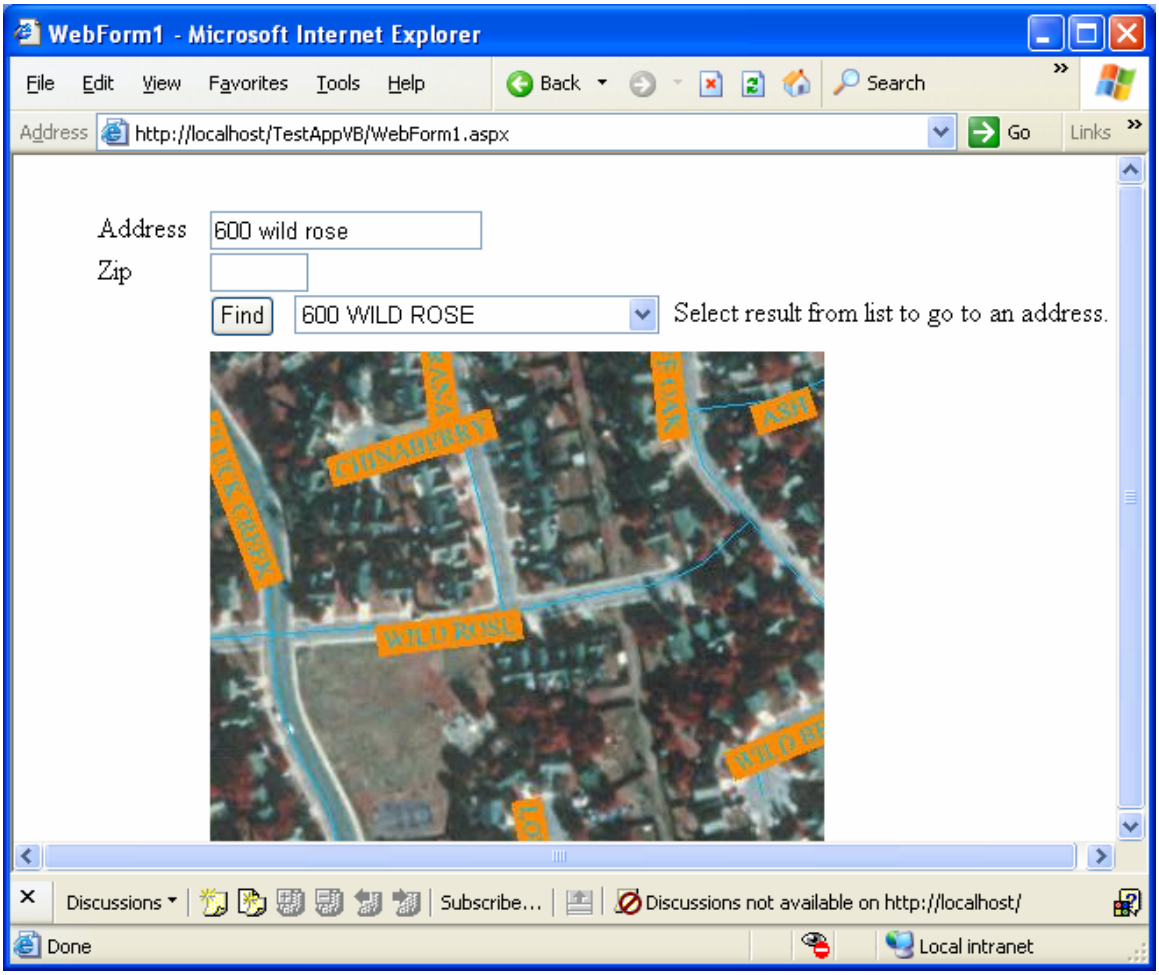
#### *4.9.1.1.3 C#*

```
private void ddAddresses_SelectedIndexChanged(object sender,
System.EventArgs e)
{
  ZoomToGeoCode((System.Data.DataTable)ViewState["__GeocodeResults"]);
}
```

#### *4.9.1.1.4 VB*

```
    Private Sub ddAddresses_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
ddAddresses.SelectedIndexChanged
        ZoomToGeoCode(CType(ViewState("__GeocodeResults"),
System.Data.DataTable))
End Sub
```

Make sure the ddAddresses control has its AutoPostBack property set to true. If not the application will not reach this code on just a change of the drop down list. Try the application now. Key an address in (600 Wild Rose) and after the initial display select another entry from the drop down. The map display will change to a location for the new address.


WebForm1 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Address
http://localhost/TestAppVB/WebForm1.aspx
Go
Links
Address
600 wild rose
Zip
Find
600 WILD ROSE
Select result from list to go to an address.
WILD ROSE
Discussions
Subscribe...
Discussions not available on http://localhost/
Done
Local intranet