How to Create, Package, and Deploy a MapXtreme 2004 Application

This white paper is designed as a tutorial to demonstrate the simplicity of creating, packaging, and deploying a MapXtreme 2004 application. At the end of this tutorial you will have gone through all the steps to successfully develop, package, and deploy a well-implemented mapping application.

This white paper assumes that you have already successfully installed Visual Studio .NET and are familiar with the use of the product. If you are not familiar with Visual Studio .NET, there are many resources available on the Microsoft MSDN web site. Look at **http://msdn.microsoft.com/vstudio/using/getstart/default.aspx**. You should also have fully installed MapXtreme 2004. All code segments described are in C#. For many of the sample applications, we provide Visual Basic .NET samples also.

In this white paper...

MapInfo.

Creating Your Application

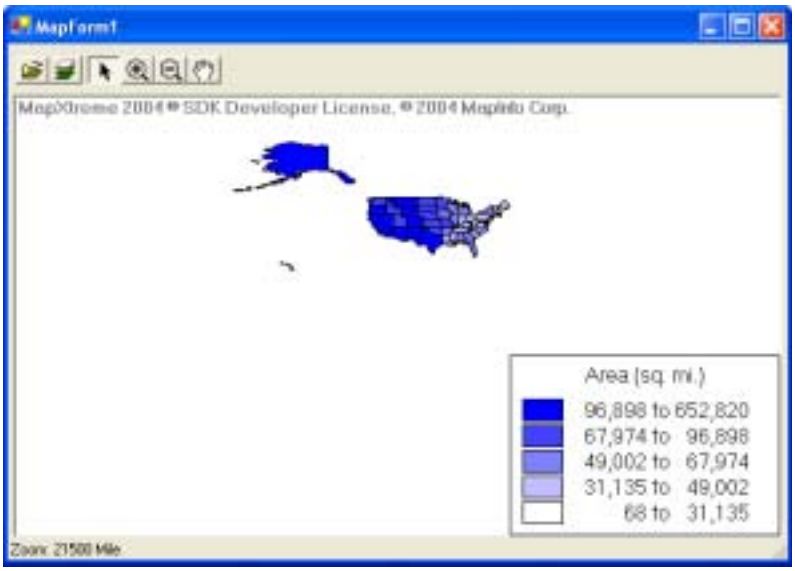
The first step is to create an application. This section will step you through this process first from a the point of view of creating a desktop application and then from the perspective of creating a web application.

Desktop Applications

1. Choose Your Template

For this example we are going to choose one of the thematics sample applications provided with MapXtreme 2004. All the sample applications provided with MapXtreme 2004 were created from the map application template provided in the product. Using Visual Studio .NET, open up the solution for the sample application called ThemeLegend (located in the Samples\Features directory in your MapXtreme 2004 installation directory). The file is C:\Program Files\MapInfo\MapXtreme\6.*x*\Samples\Features\ThemeLegend\cs\ThemeLegend.sln (where *x* is the release number of the version of MapXtreme 2004 you are using).

Once the solution is open, do a debug build by choosing **DEBUG > START** or pressing **F5**.



The map that the application creates is a standard map of the US with a theme based upon the land area of each state. The larger states are in blue and the smaller states are in white. The legend shows the breakdown. The zoom level is not very useful because the map defaults to showing the entire layer.

Use the tools provided to zoom in on a particular region, or zoom out as you choose. The tools also allow you to select a region and manipulate the way the layers are displayed using the layer control.

As you can see, building a workable application is quite simple.

What happens when the simple application is not enough and you need some more customization? The next section details some modifications that can be made to this application as a model for customizing any MapXtreme 2004 solution.

2. Modifying Your Application

Let's start by looking through the code behind the form. Using Visual Studio .NET, right click anywhere on the form and choose **VIEW CODE** to display the code page.

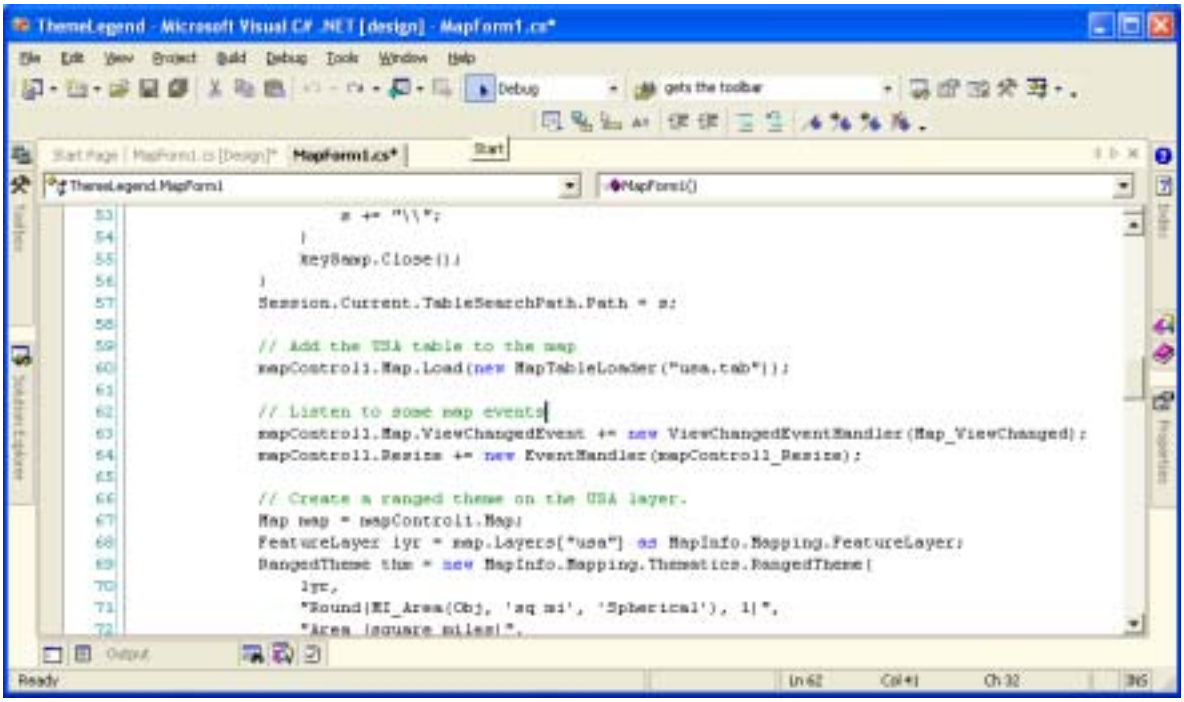
Note: All sample applications discussed in this white paper were taken from the 6.0 distribution CD of MapXtreme 2004. The line numbers referenced may not correspond exactly with the code in your installation.

As you scroll down the code page notice that the majority of the action in the application takes place in the `MapForm1()` class. The first thing we'll change is the data that the application uses to create the theme on.

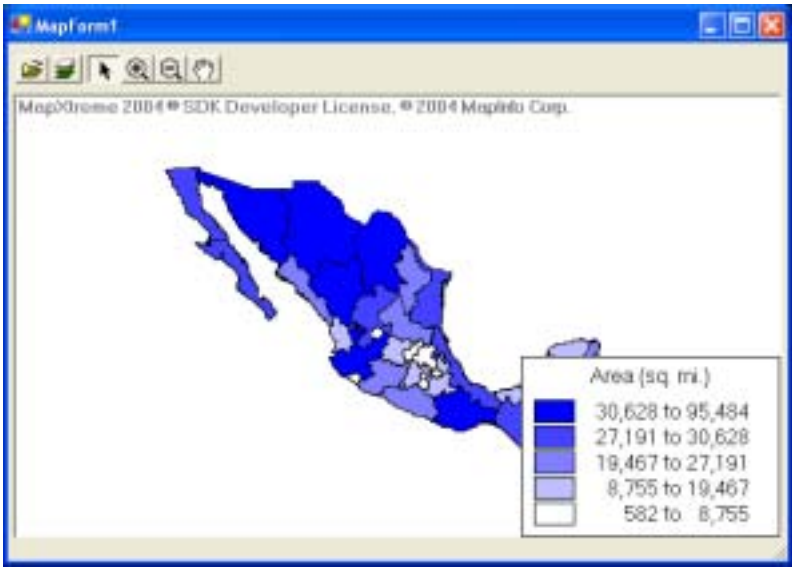
Make the following changes:

- On line 60 change `usa.tab` to `mexico.tab`
  This loads the Mexico map instead of the USA map.
- On line 68, change `usa` to `mexico`
  This changes the FeatureLayer we are working with.

Refer to the image below for guidance to the proper placement of the changed code.



Build and run this version of the application to see what changes have taken place.



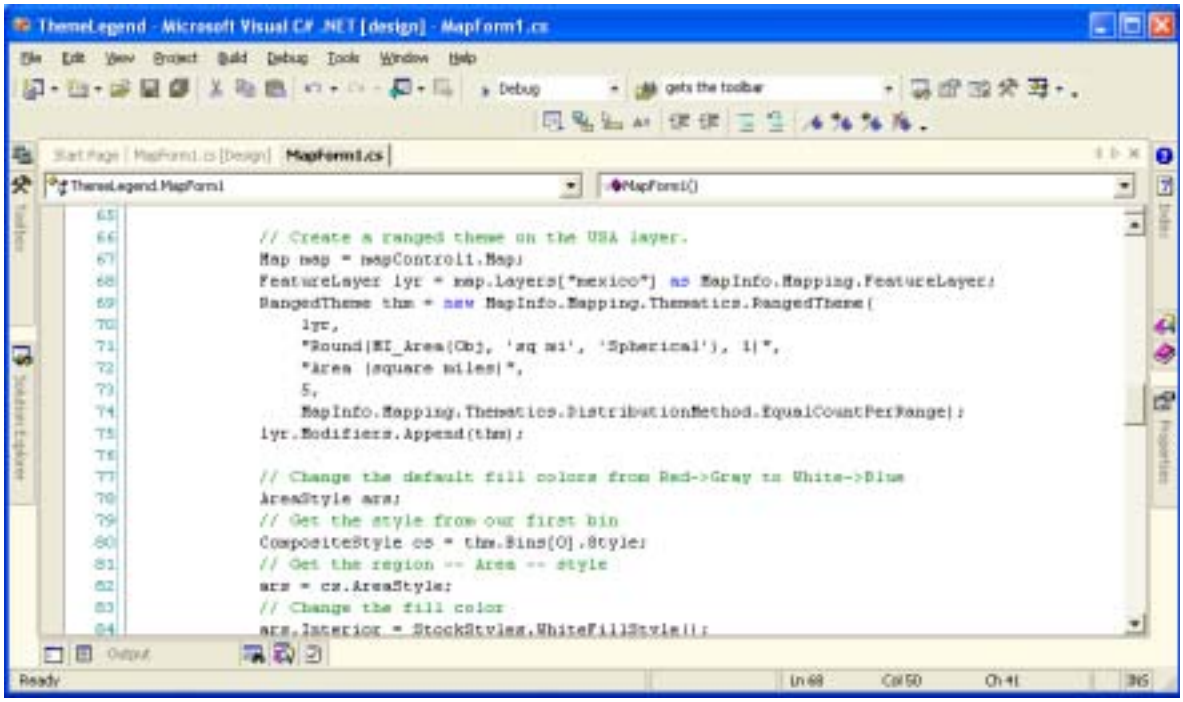
There is a clear difference in that we are using a different data source for our map. We can pass in any data source that fits this theme by changing those two lines.

Next, let's change the way the theme is displayed. By manipulating the parameters passed to the RangedTheme constructor we can change the way the theme is calculated and displayed. Place your cursor over the place in the code where the theme is created and press F1 (line 69). The help topic for this call is displayed and you can read through this to see what each parameter does when it is passed to the constructor.

Make the following changes:

- On line 73, change "5" to "8"
  This changes the number of bins for the theme. (A bin is a particular section of the data that matches a given parameter. By changing this number we will have divided the data into more "categories.")
- On line 74, change "`EqualCountPerRange`" to "`EqualRangeSize`."
  This changes the distribution method so that each range covers a similar amount of the value.

Refer to the image below for guidance of the placement of the changed code.



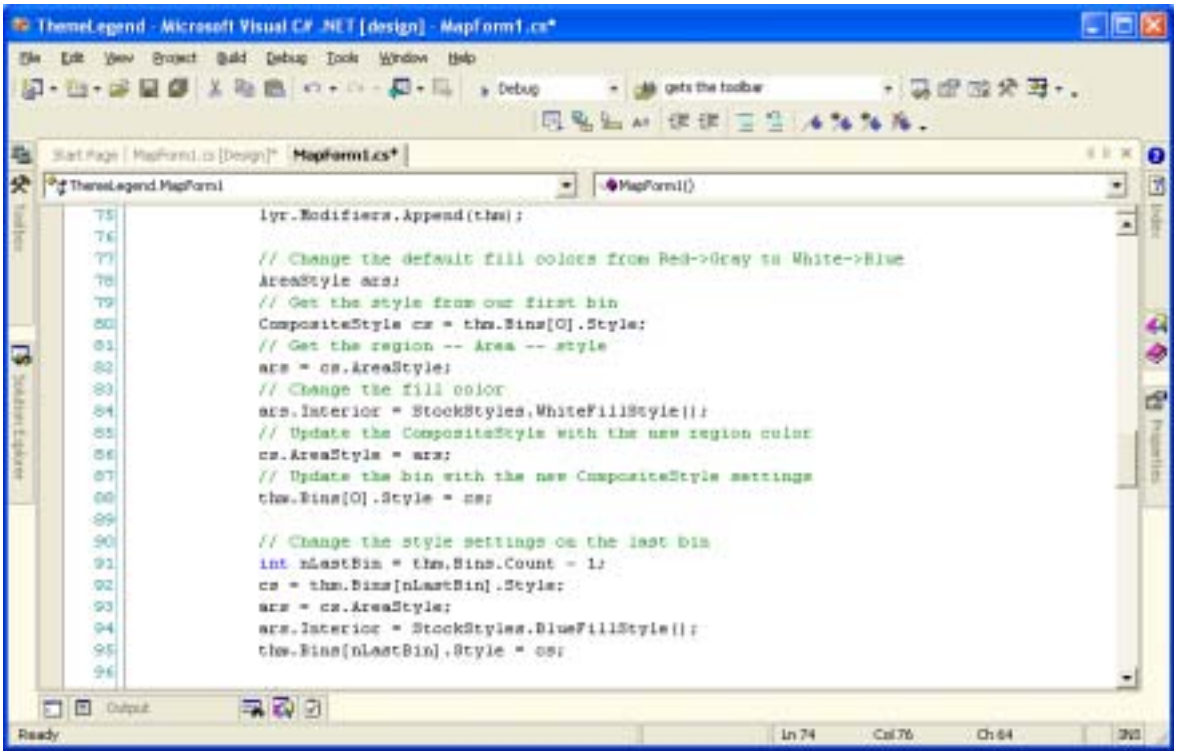
Press **F5** (or **DEBUG > START**) to build and run the modified application. As you can see, the number of theme bins has increased and the distribution has changed.

Now let's change the colors each range uses for display. We can set each bin to have a particular color or we can write the code so that the colors range from one hue to another and all we specify is the first and last color. In the code right now the color ranges from white to blue. Let's change the color to range from blue to red.

Make the following changes:

- On line 84, change `WhiteFillStyle` to `BlueFillStyle`
  This changes the color for the first bin.
- On line 94, change `BlueFillStyle` to `RedFillStyle`
  This changes the color for the last bin. Every bin in between will have a shade between the two colors.

Refer to the image below for guidance of the placement of the changed code.



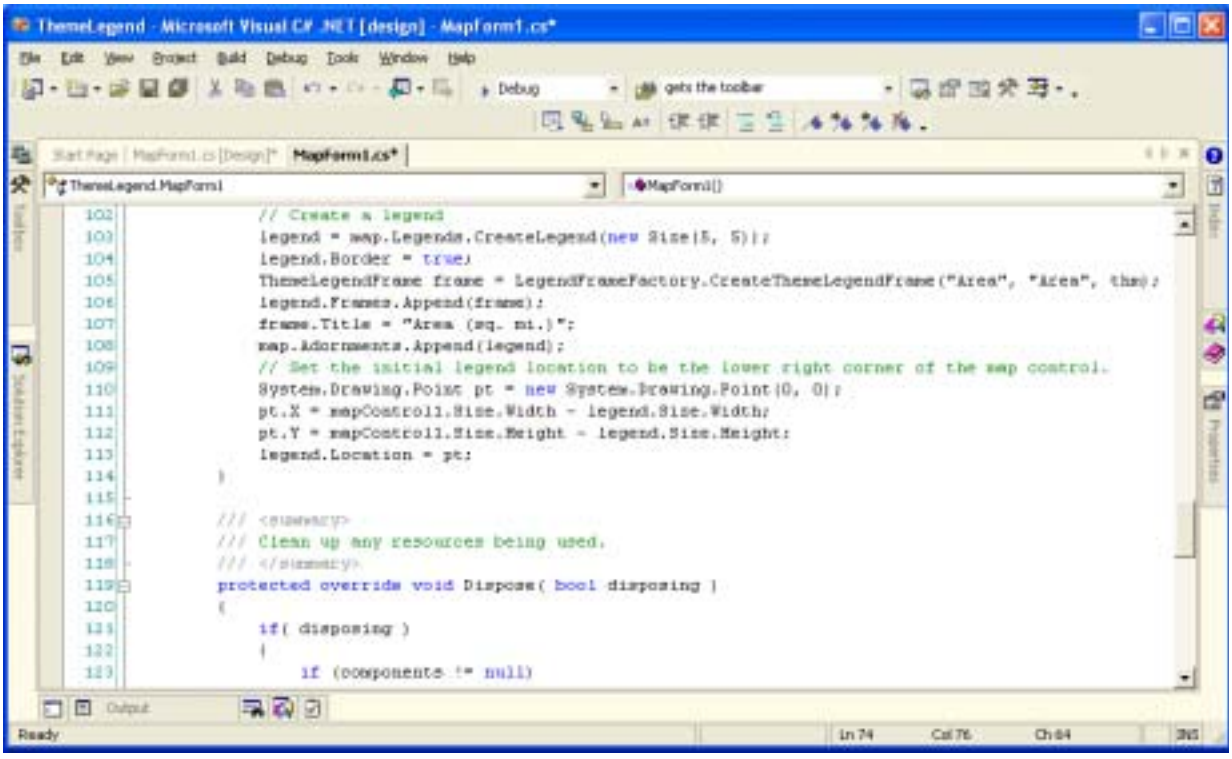
Press **F5** (or **DEBUG > START**) to build and run the modified application. As you can see, the colors of the map have changed to reflect our new settings.

Now that we have made several changes to the way the map is displayed, we need to move the legend as it is blocking some of the map that we want to see. We could use the pan tool to move the map around, but it is more efficient to move the legend programatically.

Make the following change:

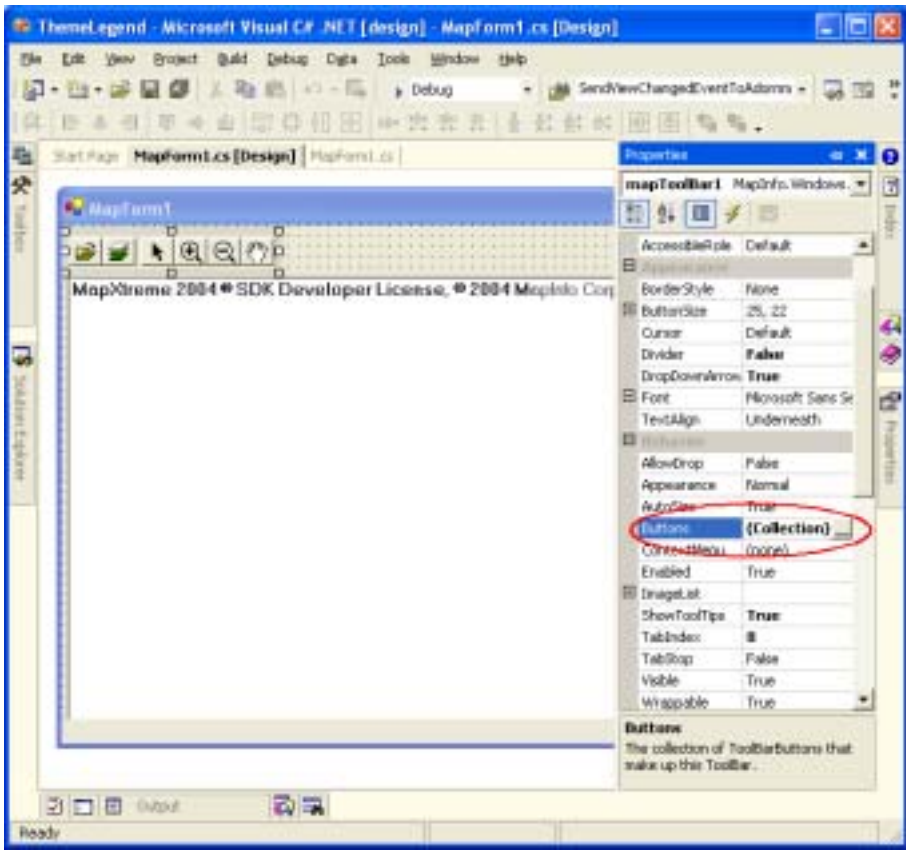
- On line 111, change "`mapControl1.Size.Width - legend.Size.Width`" to "`0`"
  This changes the X coordinate of the legend location to be at the left side of the frame. We are leaving the Y coordinate as it is.

Refer to the image below for guidance of the placement of the changed code.

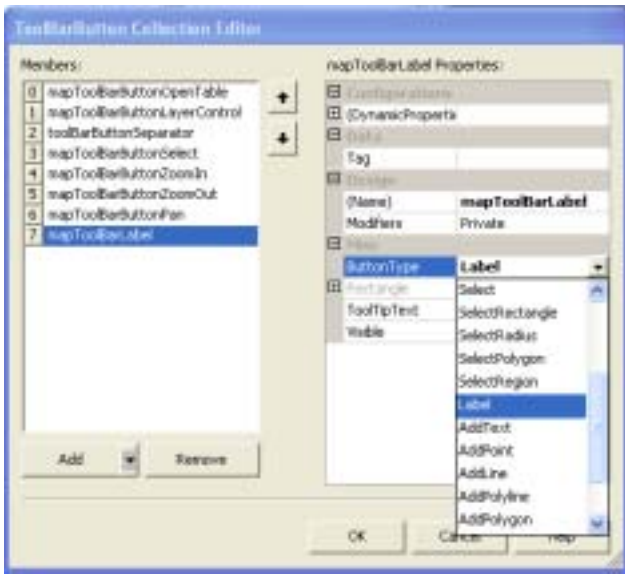


Build and run your application again to see the updated map. Now the legend is in its new location.

Let's do one more modification. This time we'll add a Label tool to the ToolBar in the interface. In Visual Studio .NET, bring the Design page to the front by clicking on the **MAPFORM1.CS [DESIGN]** tab. Click on the ellipsis (**...**) button after **COLLECTION** in the Buttons field of the Properties window.



The ToolBarButton Collection dialog box is displayed. Click the **ADD** button to add a new button to the ToolBar. This appears below the last button in the list. Rename the new button `mapToolBarLabel` in the **(NAME)** field. Change the button type to **LABEL** by choosing that from the **BUTTONTYPE** drop-down menu.



Click **OK** to save your changes and close the ToolBarButton Collection dialog box. Now you can see the new tool added to your ToolBar.

As you can see modifying an existing application is an easy way to achieve your desired results without much trouble. Other modifications we could make to this thematic map include:

- Changing the field on which to base the theme;
  By examining the table structure, you could choose population, for example. Don't forget to update the legend to match.
- Add additional themes and legends.
- Add additional tools and buttons.

3. Building Your Application

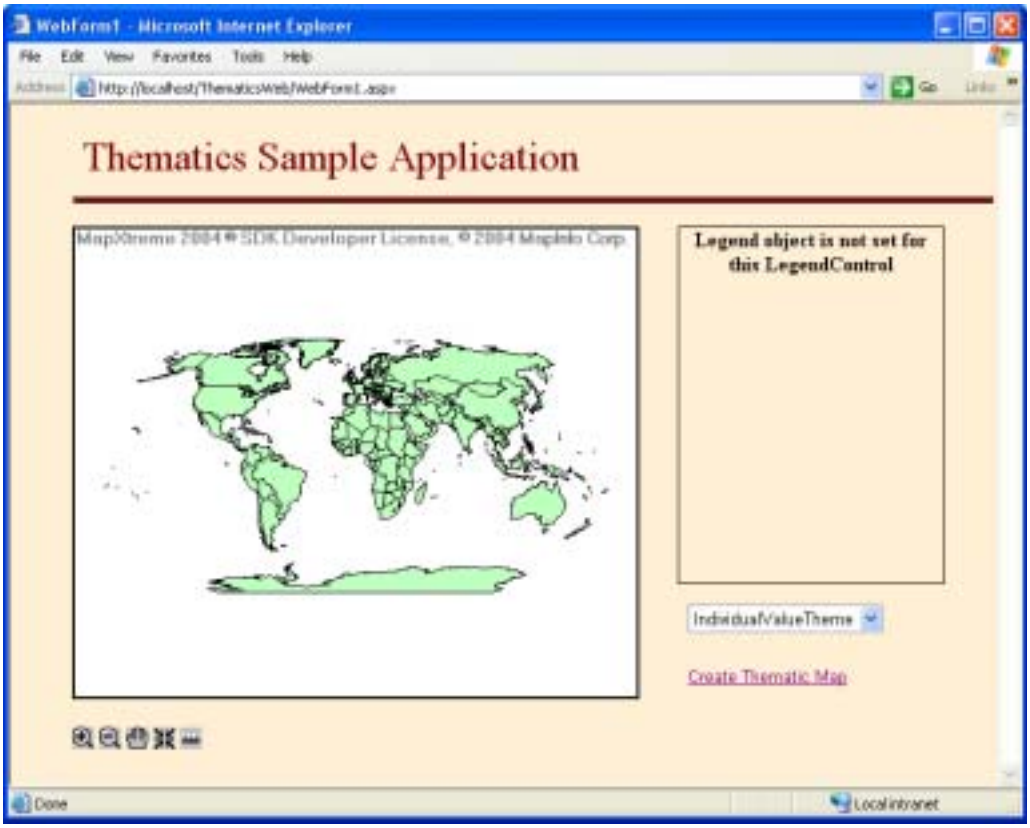
Now that you have modified your application, you are ready to do a final build. Change your build type to Release and build again.

Web Applications

1. Choose your Template

For this example we are going to choose a Web Thematics sample application that is provided with MapXtreme 2004. Open up the solution file from the sample application called ThematicsWeb (the file is C:\Program Files\MapInfo\MapXtreme\6.0\Samples\Features\ThematicsWeb\cs\ThematicsWeb.sln).

Once the solution is open, do a debug build by pressing **F5** (or **DEBUG > START**).



The web page that is displayed contains a map of the world and a pull-down menu listing different themes to be applied as well as a link that applies the themes. In the lower left-hand corner are the standard tool buttons that are included in MapXtreme 2004 web templates: ZoomIn, ZoomOut, Pan, Center, and Distance. Every theme that is added creates a legend in the LegendControl on the page. The DotDensityTheme and the GraduatedSymbol are created based upon an included MS Access database, worldcap.mdb, using the population of world capitals to determine the values.

As you can see, creating a sophisticated web application is quite simple. This particular web application is a good example of what is possible to create using MapXtreme 2004. We can modify this application to more closely match our users' needs. The next section details some modifications that can be made to this application as a model for customizing any MapXtreme 2004 web solution.

2. Modify Your Application

In this section we are going to detail modifying this application to include more data, fix a functionality anomaly, and change some of the display attributes of the themes as well as add a new one to choose from.

First, let's add in a table to be included in the display. In the desktop application we modified above, we changed the data source programatically. This time we are going to change it in the MapControl directly. Display the design page, WebForm.aspx, by clicking on that item in the Solution Explorer. Click once in the MapControl and open the properties window. At the bottom of the window click the link **CLEAR MAP** to remove all tables currently loaded. Now click the **LOAD MAP...** link to select available data sources. Choose `world.mws` for a fuller data set. As you see, this workspace includes a grid, the ocean, and a layer of world capitals.

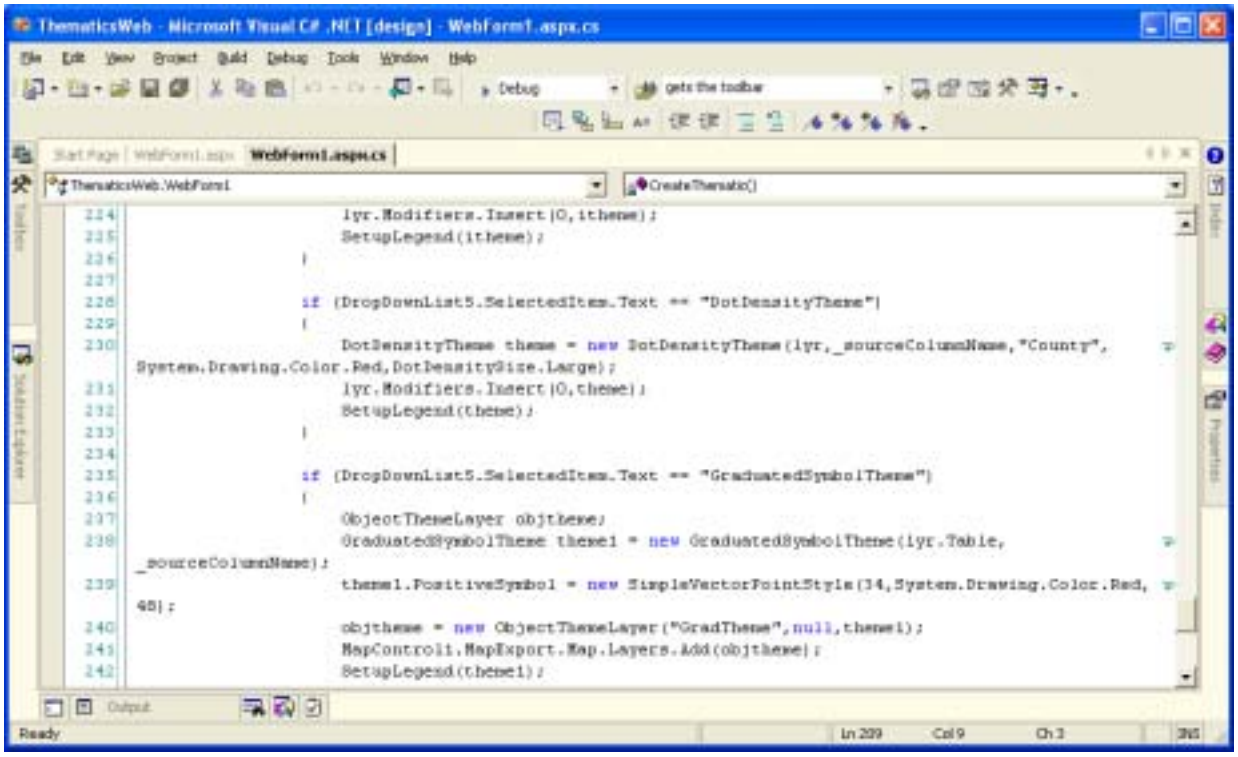
Second, let's look at the aberrant behavior of this web page. If you apply all three themes you will notice that once you apply the Graduated Symbol theme it does not get removed when applying a new theme. This can be fixed by forcing the application to remove all themes before applying a new one. Bring the code window to the front by clicking on the MapForm1.aspx.cs tab. If the tab is not visible in the Solution Explorer, right-click on WebForm1.aspx and choose **VIEW CODE**. This opens the code-behind page of the web form. Add the following code to the `LinkButton18_Click(object sender, System.EventArgs e)` method (line 278):

```
RemoveAllThemes();
```

This code calls a previously defined function that clears the map of any themes. By building and running the application again you can see that the particular problem has been fixed. Next we can modify the themes to have different colors.

`CreateThematic()` is responsible for this process and uses a series of `if` steps to achieve this. In the `if` statement that deals with the DotDensityTheme find the parameter that deals with the color of the dots. In the sample this value is currently `Red`. We can change this to any color we like from System.Drawing.Color; choose `RoyalBlue` as it shows up nicely against the green map color (line 230). Let's next change the color of the Graduated Symbols in the same manner. (On line 239) change `Red` to `Chocolate`.

Refer to the figure below for guidance on the placement of changed code.

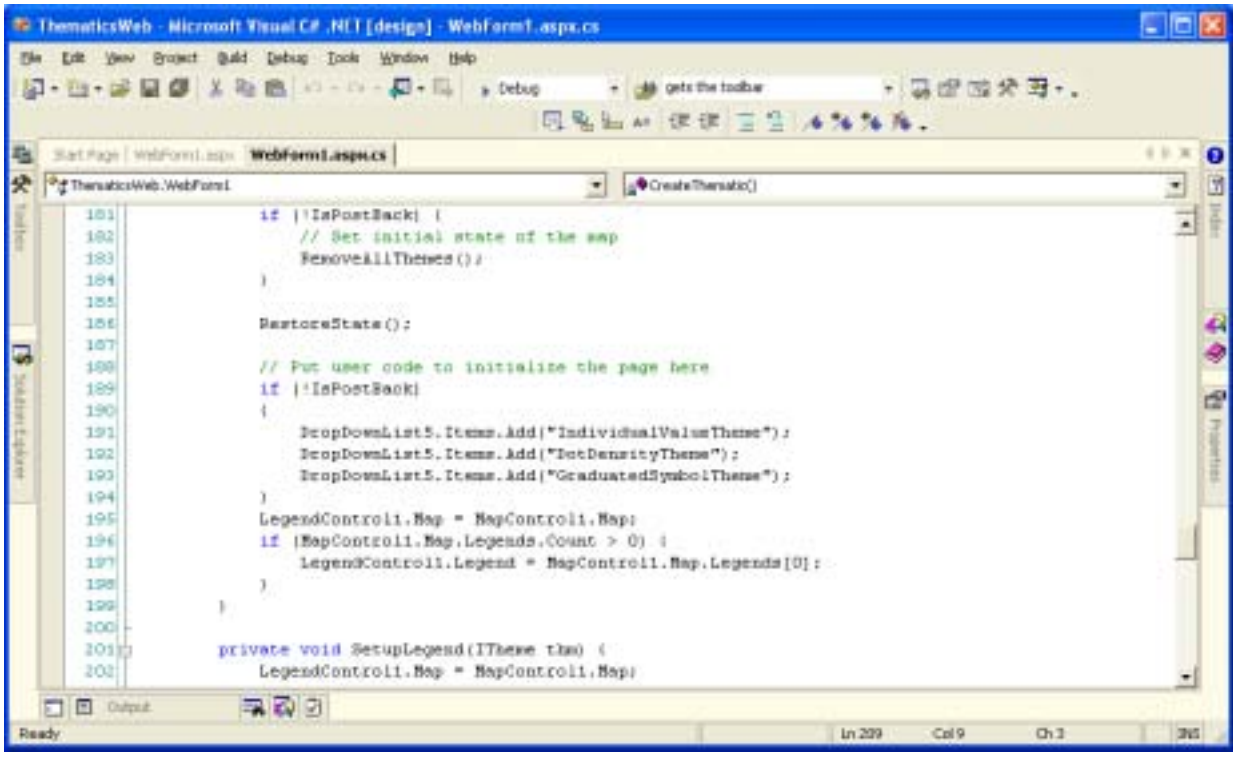


Rebuild and run your application to see the result.

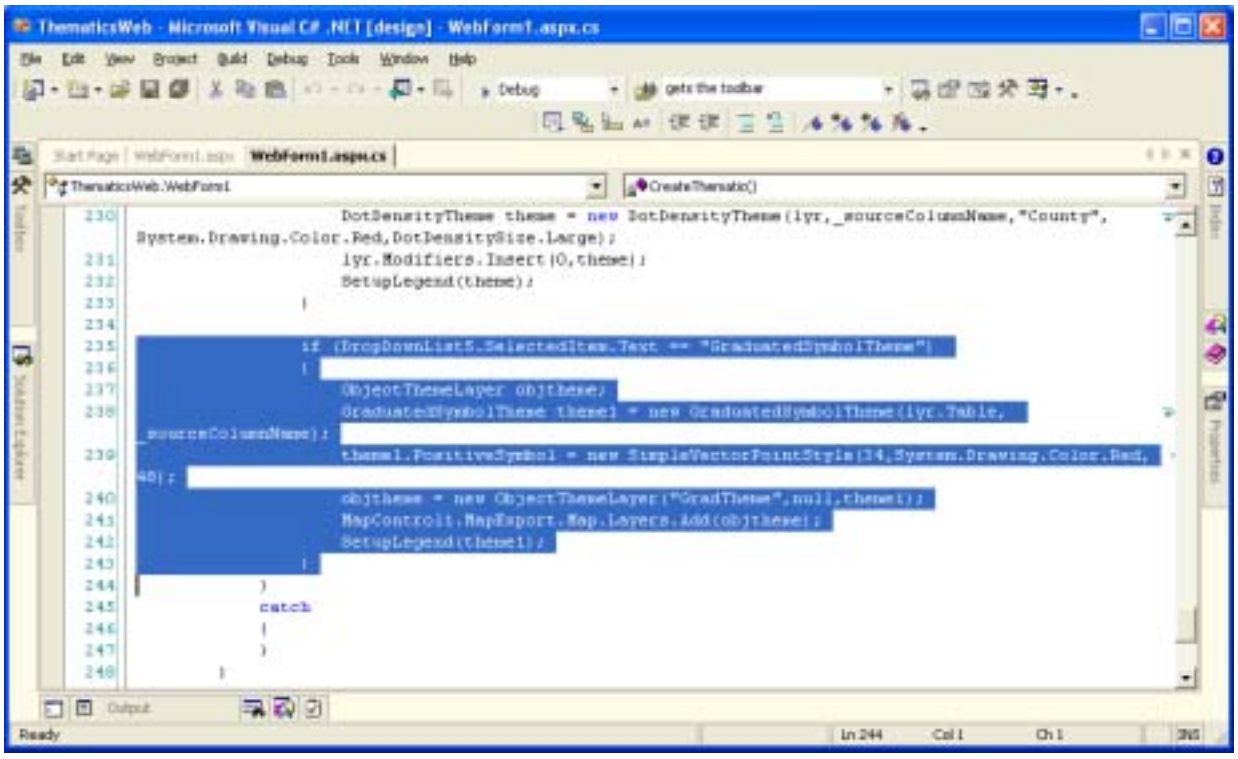
We could also add a different style of theme to the application. To do this requires two steps: 1) add the theme style to the pull-down menu and; 2) add the corresponding code to the `if` statements in `CreateThematic()`. As you can see in the code, the pull-down menu is populated programatically, rather than using the design page. To keep it simple we will add a theme that is a modification of the GraduatedSymbolTheme. We just need to add a line at the bottom the section where the list items are created. At the end of line 193 put a new line and type the following code:

```
DropDownList5.Items.Add("GraduatedFlagTheme");
```

Refer to the figure below:



Now highlight and copy the `if` statement that contains the GraduatedSymbolTheme clause (lines 235 to 243). Paste these lines below the section where you copied them. Make sure that you keep the pasted if statement outside the one above.



Change the new `if` statement as follows:

1. Change the `if` statement to read (line:

```
if (DropDownList5.SelectedItem.Text == "GraduatedFlagTheme")
```

2. Change the line where the display attributes are listed (line 248):

```
theme1.PositiveSymbol = new SimpleVectorPointStyle
                (54,System.Drawing.Color.Coral,60);
```

The changes made include changing the display symbol to a flag, the maximum display size to 60, and the color to Coral.

There are many of ways to change this application and we have only pointed out a few of them. The aim of this white paper is to show you how to modify the existing samples that we have provided to create customized applications that fit your needs.

3. Build Your Application

Now that our modifications are done, change the build option to Release and then build the entire solution again. It is in Release mode that you want to package your application, which is covered in the next section.

Packaging Your Application

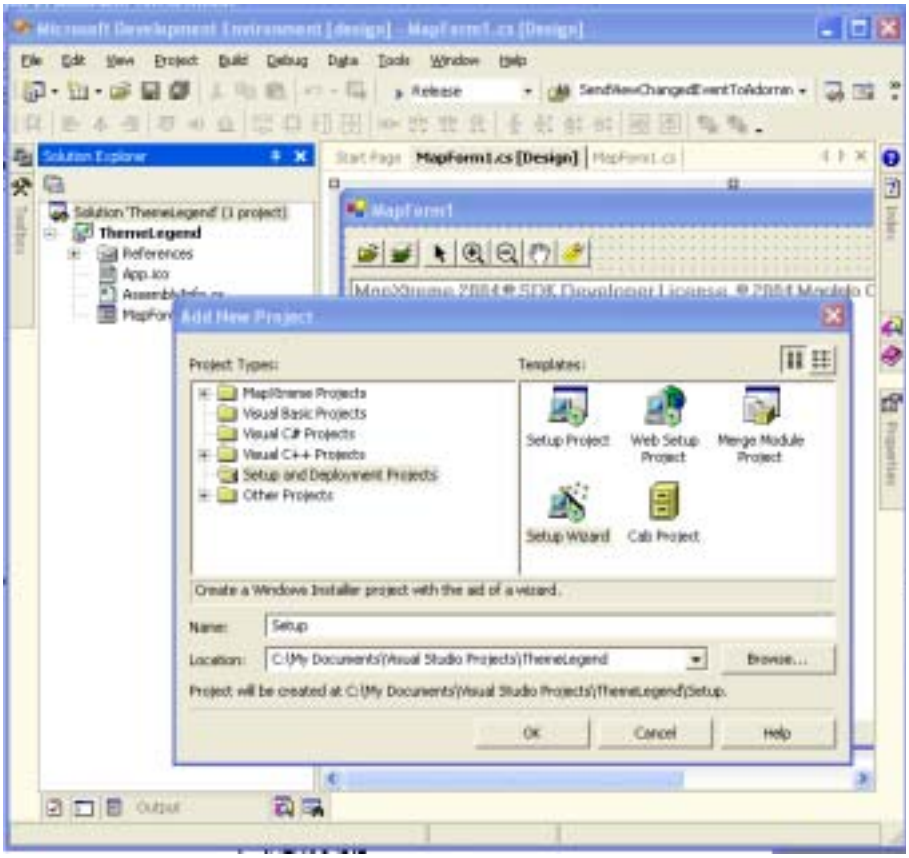
Desktop Application Packaging

Now that your application is designed, coded, and built, it is time to package it for delivery to your users. MapXtreme 2004 was designed to make this process as simple as possible. Using features in Visual Studio .NET as well as automation in MapXtreme 2004, the correct merge modules will be included in your package. A merge module (MSM) is a single package that contains all files, resources, registry entries, and setup logic necessary to install a component.

1. Create The Setup Project

Follow these steps to complete your packaging:

1. Create a new Setup project within your solution.
   a. In the Solution Explorer, highlight the Solution and choose **FILE > ADD PROJECT > NEW PROJECT...**.
   b. From Setup and Deployment Projects, select **SETUP WIZARD** which steps you through the project creation process.



c. Follow the screen prompts.

2. In step 2 of 5, select to **CREATE A SETUP FOR A WINDOWS APPLICATION**. Click **NEXT**.
3. In step 3 of 5, select the project outputs to include in your setup file.
   Choose to add Primary output. Click **NEXT**.
4. In Step 4 of 5, add in the files that contain your data. Click the **ADD** button and select your files to add to the project. In this case we are adding in the following files:
   - MEXICO.DAT
   - MEXICO.ID
   - MEXICO.IND
   - MEXICO.MAP
   - mexico.TAB

   Click the Next button to see the summary of your actions.
5. Click **FINISH** to create your setup application.
   Visual Studio .NET will then display the File System of the new setup project.

2. Add a License File

In order for your customer to be able to view your application you need to install a desktop license file to your application (MapXtremeDesktop.lic). Refer to your user's manual about acquiring additional license files for distribution. For this paper we are going to include a copy of the Trial license so you see the steps involved.

1. Activate the **FILE SYSTEM(SETUP)** tab

2. Click on the Application Folder icon.

3. Right-click and add a file. In this case we are choosing the file named MapXtremeTrial.lic (found in c:\Program Files\Common Files\MapInfo\MapXtreme\6.x). Choose MapXtremeTrial.lic in the file picker and click Open.

   The file is now added to your setup project and will be installed with the other application files into your application folder.

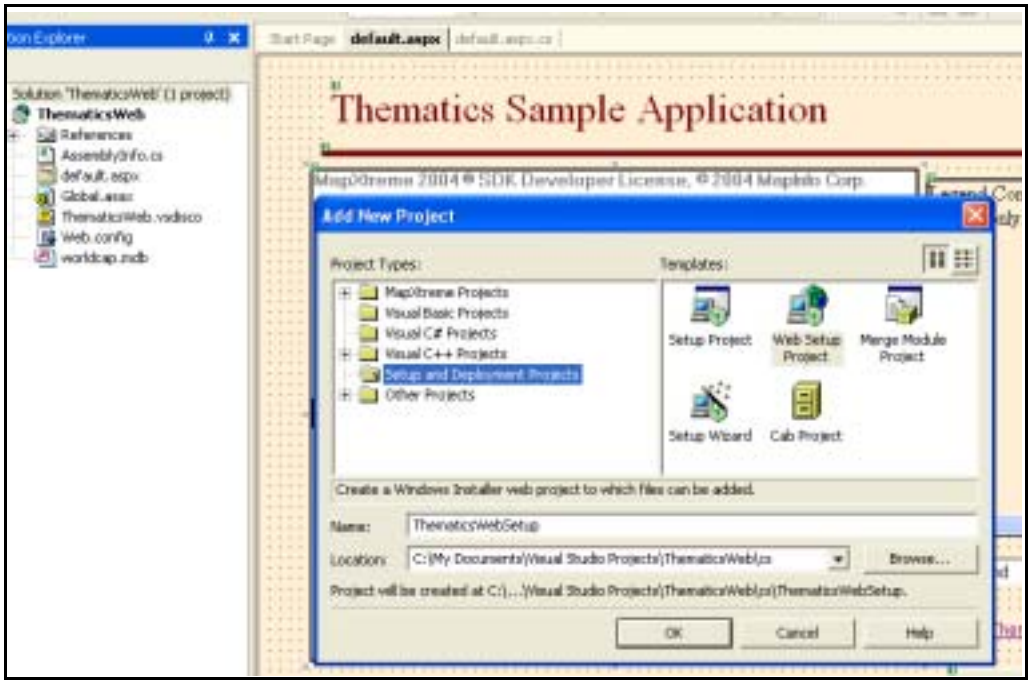
3. Build It

Now build this project and a Setup.exe file is created that contains all the data, compiled code, and included necessary MSMs for the project. This Setup project is also part of your Visual Studio .NET solution, so that when you build the entire solution, Visual Studio .NET will compile and build the mapping application and then compile and build the Setup file for installation. If you execute this setup file on another computer, the entire application, with the all the necessary files, will be installed there. That's all there is to it.

Web Application Packaging

Creating a package for a web application is similar to creating a package for a desktop application. To start we are going to add a new project to our solution. This time, instead of choosing the setup wizard, we are going to choose a Web Setup Project. Follow the steps below:

1. Create a new Setup project within your solution.
   a. In the Solution Explorer, highlight the Solution and choose **FILE > ADD PROJECT > NEW PROJECT...**.
   b. From Setup and Deployment Projects, select **WEB SETUP PROJECT** which will create a setup application specific to a web project.



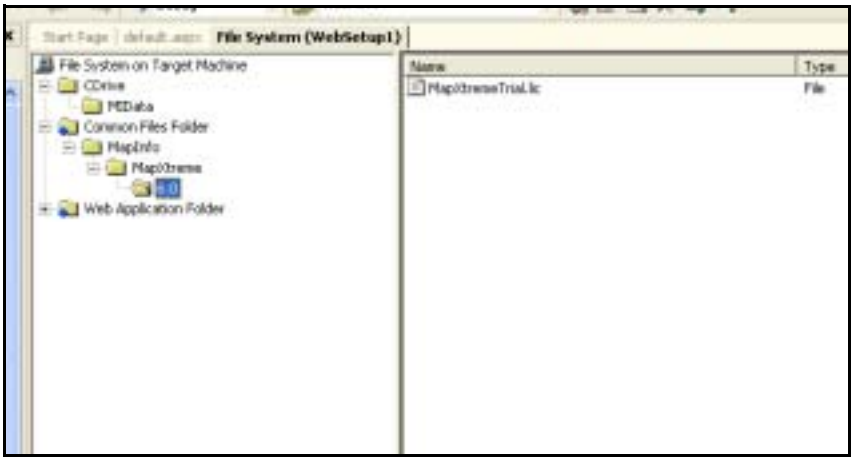
2. The next step is to indicate which parts of the solution to include. Right-click on the Setup project and choose **ADD > PROJECT OUTPUT...**. Select **PRIMARY OUTPUT** and **CONTENT FILES**.

   We need to include the content files because the web pages in our application are generated as content rather than executables as in the desktop application. Check this by looking at the properties of WebForm1.aspx. Build Action should be set to Content.

3. We need to add a local directory to hold all of our data so that the application can find it.
   a. Right-click on the setup project in the Solution Explorer and choose **VIEW > FILE SYSTEM**.
   b. In the File System window, right-click on **FILE SYSTEM ON TARGET MACHINE** and choose Add Special Folder.
   c. From the resulting list choose **CUSTOM FOLDER**. Change the name of this folder to `CDrive`. In Properties set **DEFAULTLOCATION** to `C:\`.
   d. Right-click on the CDrive directory and add a folder called MIData.
   e. Right-click the MIData folder and add in the data. The files we are including are:

GRID15.DAT	GRID15.ID	GRID15.MAP
grid15.TAB	OCEAN.DAT	OCEAN.ID
OCEAN.MAP	ocean.TAB	WLDCTY25.DAT
WLDCTY25.ID	WLDCTY25.IND	WLDCTY25.MAP
wldcty25.TAB	WORLD.DAT	WORLD.ID
WORLD.IND	WORLD.MAP	world.TAB
world.mws	WORLDCAP.DAT	WORLDCAP.ID
WORLDCAP.IND	WORLDCAP.MAP	worldcap.TAB

4. While we are looking at the File System Window, highlight on the Web Application Folder and open the Properties Window. Change the VirtualDirectory name to ThematicsWeb. This is the place where we can specify the virtual directory name for the target machine.
5. Now we need to add our license file. For web applications you need a valid copy of MapXtremeWeb.lic for your users to be able to access any maps. For this example we are using the trial license that comes with your version of MapXtreme 2004. For deployment licenses, contact your MapInfo sales associate.
   a. In the File System window, right-click on **FILE SYSTEM ON TARGET MACHINE** and choose Add Special Folder.
   b. From the resulting list choose **COMMON FILES FOLDER**.
   c. Right-click this folder and add a Folder and entitle it `MapInfo`. Continue to create nested folders for `MapXtreme` and `6.`*x* (where *x* is the release of MapXtreme 2004 you have installed) so that your file tree resembles the figure below.



d. Right-click 6.*x* and add a file. In this case we are choosing the file named MapXtremeTrial.lic (found in c:\Program Files\Common Files\MapInfo\MapXtreme\6.*x*).

6. For this application there is one more step to make sure that it works properly on an external server. We need to make the file worldcap.mdb available for the application to access as it is integral to creating some of our themes. Find this file in the Solution Explorer and highlight to examine the properties for it (if it is not visible, click the Show All Files button at the top of the Solution Explorer) .Change the setting of **BUILD ACTION** to **CONTENT**. This setting causes our installer to lay down the file in the bin directory as a file, rather than bundle it into the .dll, so that it is accessible by the application.

Okay, we're all done creating our package. Now build it by choosing **BUILD > BATCH BUILD...** and checking the Build check boxes corresponding to the Release versions of our ThematicsWeb and setup projects. Click **BUILD** and let it run. In the Release directory of our Setup project will be a file called Setup.exe and/or setup.msi. Use this file for deployment.

Deploying Your Application

Deploying Desktop Applications

Deploying a desktop application can be a trivial process. By providing the Setup.exe file that you created in the previous pages of this paper to your end user, you are, in effect, deploying the application. Simply copy the Setup.exe file to a floppy disk or CD-ROM and deliver it. When the user launches the Setup.exe executable file on their desktop computer the installation starts and steps them through the easy steps of installation. That's all there is to it!

Deploying Web Applications

To deploy a web application we go through the same steps as we did for the desktop application, transferring the file to our server and running the installation there. The setup application creates the necessary virtual directory in IIS and places all the other necessary components in the right

places. In our application, the license file is placed in the Program Files\Common Files\MapInfo\MapXtreme\6.*x* directory (where *x* is the release of MapXtreme 2004 you have installed) and a new directory is created which holds our data (C:\MIData).

To see the web application run, open a browser and navigate to the server and choose the virtual directory name. This is in the form of `http://MyMachine/ThematicsWeb/`. If you remember in **step 4** we specified this name of the virtual directory. Now you should be seeing your new web application running!

Note: Due to a problem with the Microsoft Framework you may need to end the web process, aspnet_wp.exe on your server to have your application show up with no errors. This process automatically restarts as soon as you end it, so you may notice no change in the Task Manager.

Managing State and Pooling

State and Pooling are issues for web applications to have the server and the application perform at optimum efficiency. In this paper we touch on the different topics associated with this technology, and for a deeper review of this, refer to the *MapXtreme 2004 Developers Guide*.

State

Web applications need to retain user settings from request to request. These settings can include the current map zoom, selected objects, map center, etc. Some web objects need to be retained between web requests. State is a necessary aspect of most MapXtreme 2004 applications. If you are developing a web application you need to configure how the Session is saved. The Session state can be set to either InProc, StateServer, SQLServer, or Off. When you use InProc, state is preserved using persistence and settings are kept in memory until the application is ended and you save to a workspace (.mws). If you set your Session mode to either StateServer or SQLServer, the serialization method of preserving state is used. Serialization is the conversion of the particular objects in your application to a stream of data that is stored on the server and deserialized (retrieved) when the same object is requested again by the application.

Pooling

Advantages to Using Pooling

Pooling helps reduce the number of concurrent MapInfo.Engine.ISession instances. This occurs when you have an application that has several ASP.NET sessions at the same time. Pooling helps to optimize the server's resources. The more ISession instances you have in your process the more resources are used.

A web application that has several ASP.NET requests running concurrently can use pooling to improve overall response time. By pooling the MapInfo.Engine.ISession instances, you reduce the number of requests running at the same time, which actually maximizes CPU utilization by

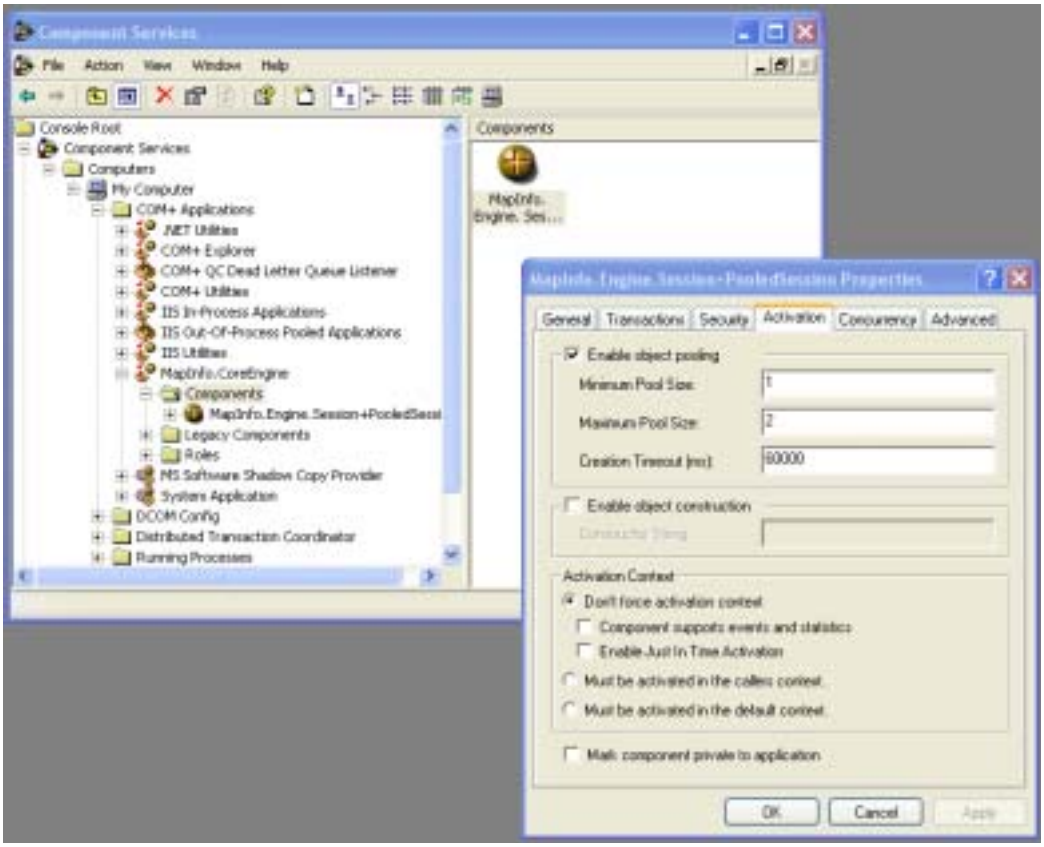
reducing thread context switching. This is particularly useful when your requests are CPU intensive (e.g., Map image exports). The general recommendation is to have a pool size of 1-2 sessions per CPU.

Configuring Pools

Pooled mode creates a pool of ISession instances. There is one pool per ASP.NET application (or AppDomain). When an ASP.NET request begins, the WebSessionActivator acquires an ISession instance from the pool. This instance is then available during the execution lifetime of the request by calling Session.Current(). When the request ends, the WebSessionActivator releases the ISession instance back into the pool. To enable pooling in your application, use the following element in your Web.config file:

```
<configuration>
   <appSettings>
      <add key="MapInfo.Engine.Session.Pooled" value="true" />
   </appSettings>
</configuration>
```

Once the pool is configured in your Web.config, open the Component Services interface of your operating system. (In Windows XP this is available through **Control Panel > Administrative Tools > Component Services**.) In Component Services navigate to **Console Root > Component Services > Computers > My Computer > COM+ Applications > MapInfo.CoreEngine > Components > MapInfo.Engine.Session+PooledSession**. Right-click on this last element and open the Properties. Choose the Activation tab and change the pool settings here.



You can configure a minimum and maximum pool size. The minimum size specifies how many ISession instances should be created when the pool is first accessed. The maximum size specifies the maximum number of pooled ISession instances to create.

This paper described several ways that you can take the provided tools in MapXtreme 2004 and modify existing sample applications to meet your own needs. As you work with MapXtreme 2004 you will find many other ways to achieve your aims as well as come up with new ways to use and display your data.The key is to continue to experiment and figure out ways to develop solutions that make your users' tasks even easier to work with their data.