

Creating Dynamic SVG images in Delphi for your web server.

Besides having straight HTML or using scripting languages like PHP or ASP, we have the ability to create CGI scripts that also can be as dynamic. CGI scripts can be compiled .exe or .dll files and can even be DOS batch files (as I found out just recently). But this document will hopefully guide you through the creation of an executable server file that generates a SVG document for your webpage.

What we will need to create this file is Delphi (which a personal version is downloadable from Borland's website) and a PC running a web server such as Microsoft's variety or what I use to develop with, Xitami. Off to the code.

Step 1. The SVG Document

What we should first produce is a working SVG document. This document will show the current day and date centered with a gradient background. I experimented with different colors and fonts until I was satisfied with my options. To determine the width of my SVG document, I chose the largest day and date that could be displayed, such as: Wednesday, October 30, 2002. This date may not actually exist, but I wanted to make sure that this information would be correctly displayed without cutting off any of the font. The following is the SVG document:

```
<?xml version="1.0" standalone="yes"?>
<svg>
<defs>
<linearGradient id="G1" x1="0" y1="0" x2="100%" y2="100%">
<stop offset="5%" stop-color="navy"/>
<stop offset="95%" stop-color="black"/>
</linearGradient>
</defs>
<rect x="0" y="0" width="260" height="25" fill="url(#G1)"
stroke="black" />
<g font-size="15" fill="yellow" stroke="green" stroke-width="0" font-
family="Verdana" >
<text x="130" y="18" text-anchor="middle" >
Monday, August 26, 2002
</text></g></svg>
```

Figure 1. Basic SVG Document.

As it stands this is a working SVG document. But what good is a static SVG document? We want to create it and never have to edit again, right?

Step 2. Running Delphi

I use Delphi 6 Professional that I bought from JourneyED. Your version may be a little different, but not much. All the following screen shots and directions will be from Delphi 6 Professional. Now run Delphi. It should, by default, create a basic form and unit. We will not need this basic windows application, so what we need to do is choose from the menu: File -> New -> Other. This will open the following window:

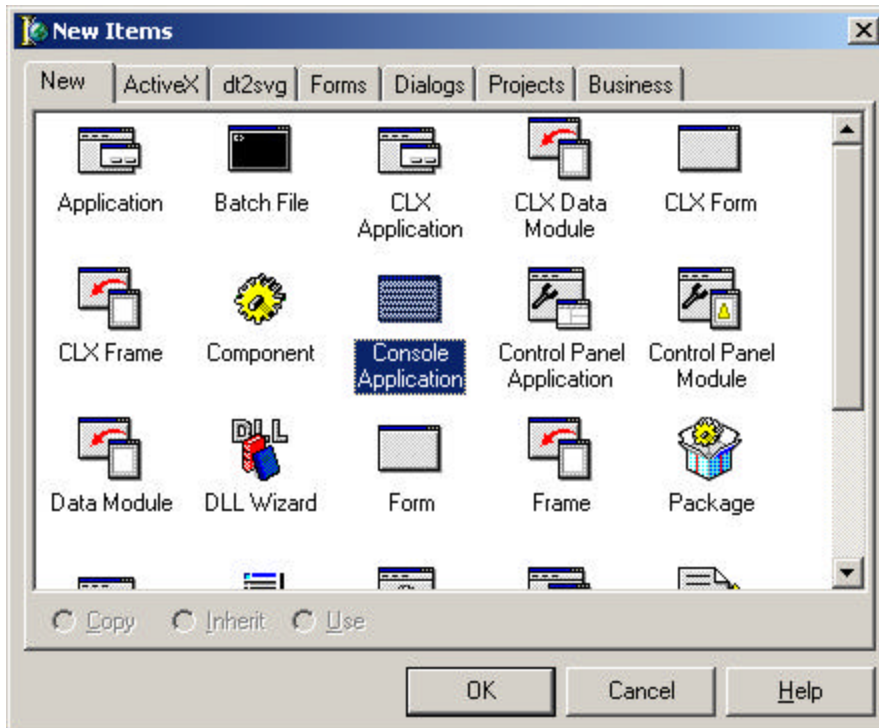


Figure 2. New Items Window.

As in Figure 2, we want to choose a Console Application. This does not have a Form and can be run from the console which is what we need. Click OK.

This will create a default Project1.dpr file and a code template.

```

Project1.dpr
Project1
program Project1;

{$APPTYPE CONSOLE}

uses
  SysUtils;

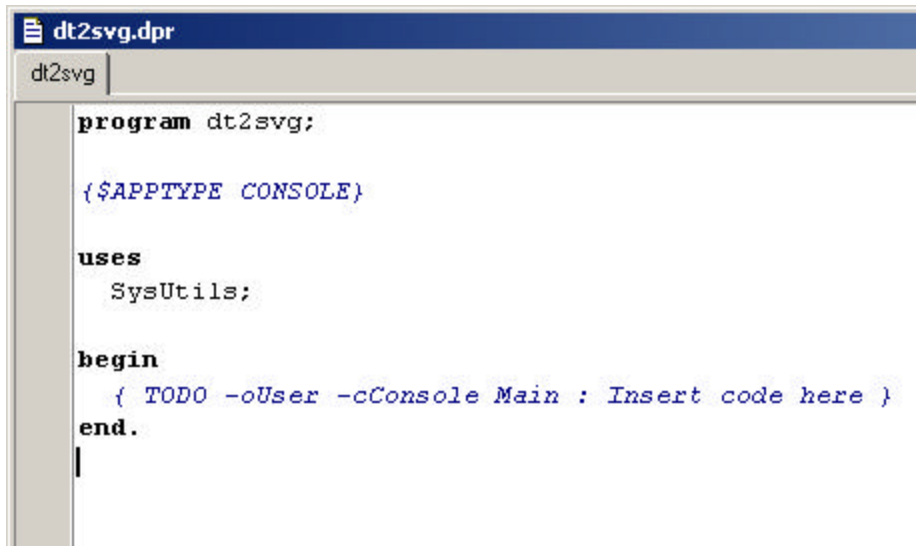
begin
  { TODO -oUser -cConsole Main : Insert code here }
end.

```

Figure 3. Code template for console application.

The opening and closing braces designate a remark. The required *{\$APPTYPE CONSOLE}* code is needed and tells the Delphi compiler that we eventually want a console program. So, we do not want to remove this line. On the other hand the second

remarked line (*TODO -oUser -cConsole Main : Insert code here*) is for our attention and can be removed later in development if desired. The code that we will add later will be added to this area between the begin and end statements. The next step is to save this application code with a different name. I chose: dt2svg.dpr (date time to svg). Your application should now resemble Figure 4.



```
dt2svg.dpr
dt2svg
program dt2svg;

{$APPTYPE CONSOLE}

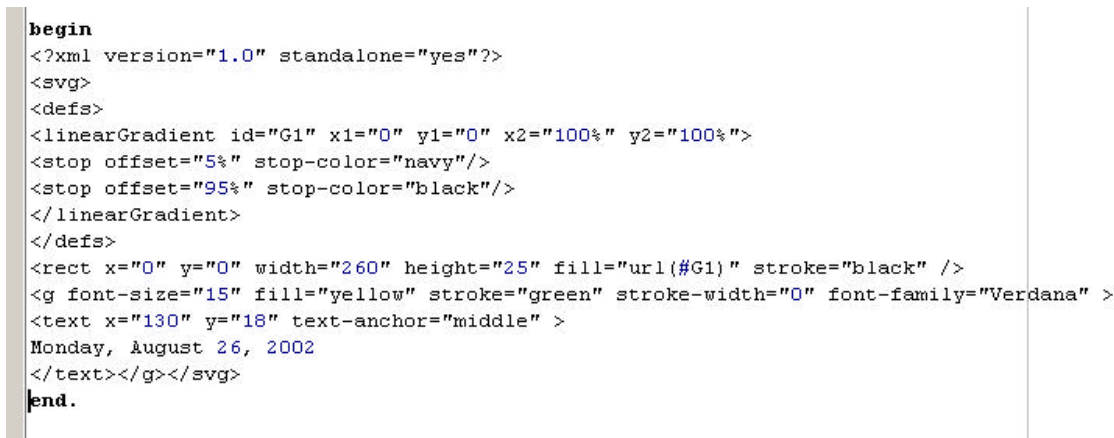
uses
  SysUtils;

begin
  { TODO -oUser -cConsole Main : Insert code here }
end.
```

Figure 4. After saving with new name.

As shown in Figure 4, the program name will change along with the tab and overall project name. We can now remove the TODO statement as we are about to enter code in this area.

Remember the SVG document code we were shown in Figure 1? Basically, we are going to cut and paste this information between the begin and end statements as shown in Figure 5.



```
begin
<?xml version="1.0" standalone="yes"?>
<svg>
<defs>
<linearGradient id="G1" x1="0" y1="0" x2="100%" y2="100%">
<stop offset="5%" stop-color="navy"/>
<stop offset="95%" stop-color="black"/>
</linearGradient>
</defs>
<rect x="0" y="0" width="260" height="25" fill="url(#G1)" stroke="black" />
<g font-size="15" fill="yellow" stroke="green" stroke-width="0" font-family="Verdana" >
<text x="130" y="18" text-anchor="middle" >
Monday, August 26, 2002
</text></g></svg>
end.
```

Figure 5. Early development.

We are now well on our way to this dynamic SVG document. Because this is straight SVG code, Delphi does not know what we want to do with this code and it is foreign to Delphi. What we need to do is add Delphi statements to the lines of SVG code.

Next we will add the Delphi statement, `writeln` to each line. This will tell the web server that we want to write each line to the console. The correct syntax for this statement is **`writeln('string');`**. So, we will add **`writeln('`** to the beginning of each line with the ending syntax of **`');`** to the end of each line. Also to make our code easy to read and change later, we will also add a couple of spaces to the beginning of each line. The updated code is shown in Figure 6.

```
begin
  writeln ('<?xml version="1.0" standalone="yes"?>');
  writeln ('<svg>');
  writeln ('<defs>');
  writeln ('<linearGradient id="G1" x1="0" y1="0" x2="100%" y2="100%">');
  writeln ('<stop offset="5%" stop-color="navy"/>');
  writeln ('<stop offset="95%" stop-color="black"/>');
  writeln ('</linearGradient>');
  writeln ('</defs>');
  writeln ('<rect x="0" y="0" width="260" height="25" fill="url(#G1)" stroke="black" />');
  writeln ('<g font-size="15" fill="yellow" stroke="green" stroke-width="0" font-family="Verdana" >');
  writeln ('<text x="130" y="18" text-anchor="middle" >');
  writeln ('Monday, August 26, 2002');
  writeln ('</text></g></svg>');
end.
```

Figure 6. Update SVG code.

Did you notice one problem? We still have the static date included in this code. The whole reason for creating this executable is that we build this file once and never have to update it everyday. So let's replace this line with a statement that will read the current day and date and write this line to the console. Replace the **`writeln ('Monday, August 26, 2002');`** with **`writeln(FormatDateTime('dddd, mmmm d, yyyy', Now));`** statement. This Delphi command tells the web server that we need to read the current day and date and format this information to be displayed in our SVG document.

Save the code. The update line is show in Figure 7.

```
writeln ('<g font-size="15" fill="yellow" stroke="green" stroke-width="0" font-family="Verdana" >');
writeln ('<text x="130" y="18" text-anchor="middle" >');
writeln (FormatDateTime('dddd, mmmm d, yyyy', Now));
writeln ('</text></g></svg>');
end.
```

Figure 7. Updated FormatDateTime statement.

Let's compile this program which will create a working executable file. Choose from the menu, Project -> Compile dt2svg or as shown in Figure 8, hit the keyboard shortcut `Ctrl+F9`.

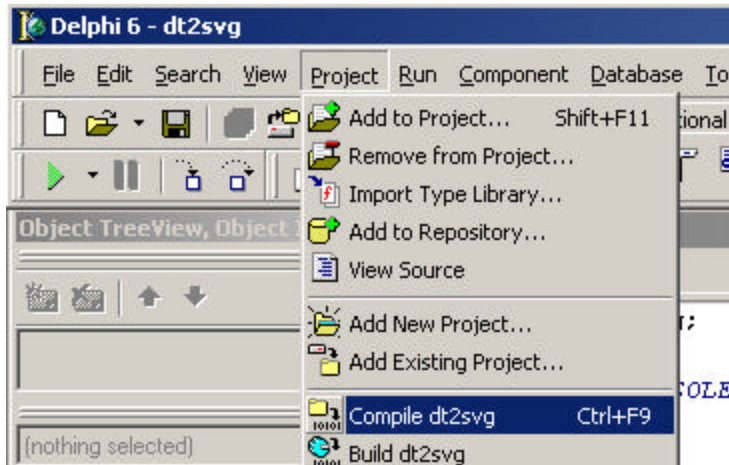


Figure 8. Compiling the executable.

You should not receive any errors during compiling and it may be so quick that you wonder if anything really happened. But, you just created dt2svg.exe! It is almost ready to be copied to your cgi scripts folder on your web server. If you are running Xitami like I am, you are basically done with the Delphi code, but if you are running any version of Microsoft's web servers there is another step that we must follow. If you are running Xitami you may skip the next part and continue on to Step 3 of this tutorial.

We need to add more lines to our code that will tell the web server what we are trying to display. Let's add two more lines to our existing code. They tell the web server that we are displaying a SVG document and not anything else such as HTML. Add **writeln('content-type: image/svg+xml');** and the **writeln;** statements to your code.

```

begin
  writeln('content-type: image/svg+xml');
  writeln;
  writeln ('<?xml version="1.0" standalone="yes"?>');
  writeln ('<svg>');
  writeln ('<defs>');
  writeln ('<linearGradient id="G1" x1="0" y1="0" x2="100%" y2="100%">');
  writeln ('<stop offset="5%" stop-color="navy"/>');
  writeln ('<stop offset="95%" stop-color="black"/>');
  writeln ('</linearGradient>');
  writeln ('</defs>');
  writeln ('<rect x="0" y="0" width="260" height="25" fill="url(#G1)" stroke="black" />');
  writeln ('<g font-size="15" fill="yellow" stroke="green" stroke-width="0" font-family="Verdana" >');
  writeln ('<text x="130" y="18" text-anchor="middle" >');
  writeln (FormatDateTime('dddd, mmmm d, yyyy', Now));
  writeln ('</text></g></svg>');
end.

```

Figure 9. Updated code for Microsoft web servers.

Compile as before.

Step 3. On to the web server with our newly created executable.

Copy the dt2svg.exe file into your web server's cgi script folder. We are now ready to create or update a document which will display this SVG information. We will use the

embed code statement along with the div statements. Edit your HTML document and add the following line(s) your HTML code.

```
<div>  
<embed src="//cgi/dt2svg.exe" name="SVGEmbed" Height="25" width="260" type="image/svg+xml">  
</noscript>  
</div>
```

Figure 10. Code added to our existing HTML document.

Notice that I kept the same height and width as our Delphi code. This is important. Also, you may have to change the location of the path to reflect where you have copied the dt2svg.exe file.

Save your HTML document. Congratulations! You have successfully created your first dynamic SVG document in Delphi that will change daily. My sample is below.

Monday, August 26, 2002

Figure 11. Completed SVG Document.

Author: Michael Keller

Platform: PC

Date: Monday, August 26, 2002