The following page will show you how to include a webserver in your .NET application, and how to let the webserver generate a dialplan.

Contents

- 1 Requirements
- 2 Implementation
- 3 Configuration
  - 3.1 See Also

Requirements

MiniHTTPd is a C# project released under the artistic license. It is coded in .NET 1.1 and will give some warnings in .NET 2.0.

It can be downloaded from here: http://www.codeproject.com/KB/IP/minihttpd.aspx

Implementation

First of all, you need to download and install MiniHTTPd. The next thing to do is to create a new project (whatever flavor you like), and add this class to it.

```
using System;
using System.Collections.Generic;
using System.Text;
using System.IO;
using MiniHttpd;

namespace FreeSwitch
{
    /// <summary>
    /// The following class is coded by Jonas Gauffin.
    ///
    /// </summary>
    class FreeSwitchCurlPage : IFile
    {
        private IDirectory parent = null;
        public delegate string HandleQuery(UriQuery query);
        public event HandleQuery OnGetDialplan;
        public event HandleQuery OnGetUserDirectory;

        public FreeSwitchCurlPage()
        {
        }
        public FreeSwitchCurlPage(IDirectory parent)
        {
            this.parent = parent;
        }
```

```
        /// <summary>
        /// Gets the name of the entry.
        /// </summary>
        public string Name
        {
            get { return "curl.fetch"; }
        }

        /// <summary>
        /// Gets the parent directory of the object.
        /// </summary>
        public IDirectory Parent
        {
            get { return parent; }
        }

        /// <summary>
        /// Called when the file is requested by a client.
        /// </summary>
        /// <param name="request">The <see cref="HttpRequest"/> requesting the file.</param>
        /// <param name="directory">The <see cref="IDirectory"/> of the parent directory.</param>
        public void OnFileRequested(HttpRequest request, IDirectory directory)
        {
            UriQuery queryString = new UriQuery(GetPostData(request));
            if (queryString["section"] == "dialplan")
            {
                if (OnGetDialplan != null)
                {
                    string dialplan = OnGetDialplan.Invoke(queryString);
                    request.Response.BeginChunkedOutput();
                    System.IO.StreamWriter writer = new StreamWriter(request.Response.ResponseCon
                    writer.Write(dialplan);
                    writer.Flush();
                    writer.Close();
                }
            }
            else if (queryString["section"] == "directory")
            {
                if (OnGetUserDirectory != null)
                {
                    string userDirectory = OnGetUserDirectory.Invoke(queryString);
                    request.Response.BeginChunkedOutput();
                    System.IO.StreamWriter writer = new StreamWriter(request.Response.ResponseCon
                    writer.Write(userDirectory);
                    writer.Flush();
                    writer.Close();
                }

            }
        }

        /// <summary>
        /// Gets the MIME type of the content.
        /// </summary>
        public string ContentType
        {
            get
            {
                return "text/xml";
            }
        }
```

```
        public void Dispose()
        {
        }

        void SetParent(IDirectory parent)
        {
            this.parent = parent;
        }


        private string GetPostData(HttpRequest request)
        {
            try
            {
                using (StreamReader reader = new StreamReader(request.PostData))
                {
                    return reader.ReadToEnd();
                }
            }
            catch (ArgumentException err)
            {
                Console.WriteLine(err.ToString());
            }
            return "";
        }
    }
}
```

Next, you should add the following class which setups MiniHTTPd with our own page handler.

```
using System;
using MiniHttpd;
using System.IO;

namespace FreeSwitch
{
        /// <summary>
        /// Summary description for HttpTest.
        /// </summary>
        public class ModCurlHandler
        {
        private HttpWebServer server = null;

        public ModCurlHandler()
                {
            server = new HttpWebServer(8080);

            // Setup directories
            VirtualDirectory root = new VirtualDirectory("/", null);
            server.Root = root;

            // We want curl to call http://localhost:8080/freeswitch/curl.fetch
            VirtualDirectory dir = new FreeSwitchVirtualDirectory("freeswitch", root);
            FreeSwitchCurlPage curlPage = new FreeSwitchCurlPage(dir);
            curlPage.OnGetDialplan += OnGetDialplan;
            dir.AddFile(curlPage);
            root.AddDirectory(dir);
                }

        public void Start()
        {
```

```
            server.Start();
        }

        public void Stop()
        {
            server.Stop();
        }

        protected string OnGetDialplan(UriQuery query)
        {
            return "<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?>\r\n" +
                    "<document type=\"freeswitch/xml\">\r\n" +
                    "  <section name=\"dialplan\" description=\"RE Dial Plan For FreeSwitch\">\r\
                    "    <context name=\"default\">\r\n" +
                    "      <extension name=\"test9\">\r\n" +
                    "        <condition field=\"destination_number\" expression=\"^1111$\">\r\n"
                    "          <action application=\"playback\" data=\"C:\\temp\\sounds\\tt-monke
                    "        </condition>\r\n" +
                    "      </extension>\r\n" +
                    "    </context>\r\n" +
                    "  </section>\r\n" +
                    "</document>\r\n";
        }
        }
}
```

You should change the OnGetDialplan to do something useful=) For instance, fetch actions from the database or something like that.

Configuration

Next, you need to configure mod_xml_curl to fetch the dialplan from our code. That configuration should look like this:

```
<configuration name="xml_curl.conf" description="cURL XML Gateway">
  <bindings>
    <binding name="dialplan fetcher">
      <param name="gateway-url" value="http://localhost:8080/freeswitch/curl.fetch" bindings="dia
    </binding>
  </bindings>
</configuration>
```

See Also

- mod_xml_curl
- Mod xml curl Ruby dialplan example