URIService Behind Hostname Separated NGINX Server Blocks

If you have an API service that is called by multiple server blocks of an NGINX proxy, and the server_name is different between each one, then your NGINX is using hostname separation to identify what origin is used.

When this happens, the origin (scheme, host, and port) passed from NGINX to your API service will not have the correct origin data, for composing URLs that redirect to other endpoints in the API.

So, a singleton registered URIService instance, which is commonly setup in Startup.cs, cannot be used.

To derive a proper URIService instance, you will have to create it on-the-fly, or cache a set of singletons that are keyed by origin string.

The method you choose may depend on usage.

> NOTE: This implementation depends on the NGINX proxy forwarding schema, host, and port to your service, and your service explicitly accepting these forwarded headers.
>
> See this page for how to implement that: Getting Correct Scheme, Host, Port Behind a Proxy

But, below is a helper class that will correctly create an instance of URIService from a given request.

```
using Microsoft.AspNetCore.Http;
using OGA.InfraBase.Services;
using System;
using System.Text;

namespace OGA.Groundplane.Service
{
   static public class Helpers
   {
```

```
/// <summary>
/// Returns a URI Service instance that is correctly assigned to the origin hostname of the given request.
/// We use this, because our service can be called from multiple subdomains,
///  and the default URI Service only works with one, as it is a singleton.
/// </summary>
/// <param name="httpContext"></param>
/// <returns></returns>
static public UriService Get_URIService(HttpContext httpContext)
{
    HttpRequest request = httpContext!.Request;

    // Dump headers to log...
    if(OGA.SharedKernel.Logging_Base.Logger_Ref?.IsDebugEnabled ?? false)
    {
        var b = new StringBuilder();
        foreach (var header in request.Headers)
            b.AppendLine($"{header.Key}: {header.Value}");
        OGA.SharedKernel.Logging_Base.Logger_Ref?.Debug(
            $"Current request headers are:\n" + b.ToString());
    }

    // Retrieve the scheme that was forwarded by NGINX...
    // This is set by the following directive in NGINX:
    //  proxy_set_header X-Forwarded-Proto $scheme;
    var scheme = request.Scheme;
    // Retrieve the host that was forwarded by NGINX...
    // This is set by the following directive in NGINX:
    //  proxy_set_header Host $host;
    var hostname = request.Host.Host;
    // Retrieve the port that was forwarded by NGINX...
    // This is set by the following directive in NGINX:
    //  proxy_set_header X-Forwarded-Port $server_port;
    var port = request.Host.Port;

    OGA.SharedKernel.Logging_Base.Logger_Ref?.Debug(
        $"Current request has scheme ({scheme}), host ({hostname}), port ({(port ?? -1)})");

    // In case the port was not set, we will accept defaults, based on scheme...
    if (request.Host.Port.HasValue)
        port = request.Host.Port.Value;
```

```
            else
            {
                if (scheme == "http")
                    port = 80;
                else if (scheme == "https")
                    port = 443;
            }

            bool nondefaultport = false;
            if(scheme == "http" && port != 80)
                nondefaultport = true;
            if(scheme == "https" && port != 443)
                nondefaultport = true;

            OGA.SharedKernel.Logging_Base.Logger_Ref?.Debug(
                $"Building URI Service from these: scheme ({scheme}), host ({request.Host.ToUriComponent()}).");

            // Compose the base url string...
            string baseUri = scheme + "://" + hostname;
            // If the port is not default, add it explicitly...
            if(nondefaultport)
                baseUri = baseUri + ":" + port.ToString();

            // Create the uri service from the above base url string...
            var svc = new UriService(baseUri);

            OGA.SharedKernel.Logging_Base.Logger_Ref?.Debug(
                "Current URIService baseURL is:" + svc.Compose_Url_to_Route("").ToString());

            return svc;
        }
    }
}
```

Revision #3
Created 29 September 2025 00:50:26 by glwhite
Updated 29 September 2025 01:05:32 by glwhite