You have reached the cached page for
https://gist.github.com/maartenba/77ca6f9cfef50efa96ec
Below is a snapshot of the Web page as it appeared on 5/5/2015 (the last time our crawler visited it). This is the version of the page that was used for ranking your search results. The page may have changed since we last cached it. To see what might have changed (without the highlights), go to the current page.
You searched for: DomainTemplateRoute.cs We have highlighted matching words that appear in the page below.
Bing is not responsible for the content of this page.
Sign up for a GitHub account Sign in

All Gists
maartenba / DomainTemplateRoute - GetVirtualPath
Last active February 17, 2015

Code
Revisions 4
Stars 2
Forks 2
Embed URL

HTTPS clone URL

SSH clone URL

You can clone with HTTPS or SSH.

Download Gist
ASP.NET MVC 6 / ASP.NET 5 Domain Routing + Tenant Middleware
View DomainTemplateRoute - GetVirtualPath
DomainTemplateRoute - GetVirtualPath
Raw
File suppressed. Click to show.

```
1. 2 3 4 5 6 7 8 9
public string GetVirtualPath(VirtualPathContext context)
{
    foreach (var matcherParameter in _matcher.Template.Parameters)
    {
        context.Values.Remove(matcherParameter.Name); // make sure
        none of the domain-placeholders are appended as query string
        parameters
    }

    return _innerRoute.GetVirtualPath(context);
}
```

View DomainTemplateRoute - GetVirtualPath
DomainTemplateRoute - RouteAsync
Raw
File suppressed. Click to show.

```
1. 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
   26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42
public async Task RouteAsync(RouteContext context)
{
    EnsureLoggers(context.HttpContext);
    using (_logger.BeginScope("DomainTemplateRoute.RouteAsync"))
    {
        var requestHost = context.HttpContext.Request.Host.Value;
        if (IgnorePort && requestHost.Contains(":")) // check if we
        want to match a port as well
        {
            requestHost = requestHost.Substring(0,
            requestHost.IndexOf(":"));
        }

        var values = _matcher.Match(requestHost);
        if (values == null)
        {
            // if we got back a null value set, that means the URI
            did not match
            return;
        }

        var oldRouteData = context.RouteData;

        var newRouteData = new RouteData(oldRouteData);
        MergeValues(newRouteData.DataTokens, DataTokens);
        newRouteData.Routers.Add(_target);
        MergeValues(newRouteData.Values,
        values.ToImmutableDictionary());

        try
        {
            context.RouteData = newRouteData;

            // delegate further processing to inner route
            await _innerRoute.RouteAsync(context);
        }
        finally
        {
            // Restore the original values to prevent polluting the
            route data.
            if (!context.IsHandled)
            {
                context.RouteData = oldRouteData;
            }
        }
    }
}
```

View DomainTemplateRoute - GetVirtualPath
DomainTemplateRoute.cs
Raw
File suppressed. Click to show.

```
1. 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
   26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
   48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69
   70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91
   92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109
   110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
   126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141
   142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157
   158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173
using System;
using System.Collections.Generic;
using System.Collections.Immutable;
using System.Threading.Tasks;
using Microsoft.AspNet.Http;
using Microsoft.AspNet.Routing;
using Microsoft.AspNet.Routing.Template;
using Microsoft.Framework.DependencyInjection;
using Microsoft.Framework.Logging;

namespace Multitenancy.Routing
{
    public class DomainTemplateRoute
        : INamedRouter, IRouter
    {
        private readonly TemplateRoute _innerRoute;
        private readonly IRouter _target;
        private readonly string _domainTemplate;
        private readonly TemplateMatcher _matcher;
        private ILogger _logger;

        public DomainTemplateRoute(IRouter target, string
        domainTemplate, string routeTemplate, bool ignorePort,
        IInlineConstraintResolver inlineConstraintResolver)
            : this(target, domainTemplate, routeTemplate, null,
            null, null, ignorePort, inlineConstraintResolver)
        {
        }

        public DomainTemplateRoute(IRouter target, string
        domainTemplate, string routeTemplate, IDictionary<string,
        object> defaults, IDictionary<string, object> constraints,
        IDictionary<string, object> dataTokens, bool ignorePort,
        IInlineConstraintResolver inlineConstraintResolver)
            : this(target, null, domainTemplate, routeTemplate,
            defaults, constraints, dataTokens, ignorePort,
            inlineConstraintResolver)
        {
        }

        public DomainTemplateRoute(IRouter target, string routeName,
        string domainTemplate, string routeTemplate,
        IDictionary<string, object> defaults, IDictionary<string,
        object> constraints, IDictionary<string, object> dataTokens,
```

```
bool ignorePort, IInlineConstraintResolver
inlineConstraintResolver)
{
    _innerRoute = new TemplateRoute(target, routeName,
    routeTemplate, defaults, constraints, dataTokens,
    inlineConstraintResolver);

    _target = target;
    _domainTemplate = domainTemplate;

    _matcher = new TemplateMatcher(
        TemplateParser.Parse(DomainTemplate), Defaults);

    Name = routeName;
    IgnorePort = ignorePort;
}

public string Name { get; private set; }

public IReadOnlyDictionary<string, object> Defaults
{
    get
    {
        return _innerRoute.Defaults;
    }
}

public IReadOnlyDictionary<string, object> DataTokens
{
    get
    {
        return _innerRoute.DataTokens;
    }
}

public string RouteTemplate
{
    get
    {
        return _innerRoute.RouteTemplate;
    }
}

public IReadOnlyDictionary<string, IRouteConstraint>
Constraints
{
    get
    {
        return _innerRoute.Constraints;
    }
}

public string DomainTemplate
{
```

```
    get
    {
        return _domainTemplate;
    }
}

public bool IgnorePort { get; set; }

public async Task RouteAsync(RouteContext context)
{
    EnsureLoggers(context.HttpContext);
    using
    (_logger.BeginScope("DomainTemplateRoute.RouteAsync"))
    {
        var requestHost =
        context.HttpContext.Request.Host.Value;
        if (IgnorePort && requestHost.Contains(":"))
        {
            requestHost = requestHost.Substring(0,
            requestHost.IndexOf(":"));
        }

        var values = _matcher.Match(requestHost);
        if (values == null)
        {
            if (_logger.IsEnabled(LogLevel.Verbose))
            {
                _logger.WriteVerbose("DomainTemplateRoute "
                + Name + " - Host \"" +
                context.HttpContext.Request.Host + "\" did
                not match.");
            }

            // If we got back a null value set, that means
            the URI did not match
            return;
        }

        var oldRouteData = context.RouteData;

        var newRouteData = new RouteData(oldRouteData);
        MergeValues(newRouteData.DataTokens, DataTokens);
        newRouteData.Routers.Add(_target);
        MergeValues(newRouteData.Values,
        values.ToImmutableDictionary());

        try
        {
            context.RouteData = newRouteData;

            // delegate further processing to inner route
            await _innerRoute.RouteAsync(context);
        }
        finally
```

```
            {
                // Restore the original values to prevent
                polluting the route data.
                if (!context.IsHandled)
                {
                    context.RouteData = oldRouteData;
                }
            }
        }
    }

    public string GetVirtualPath(VirtualPathContext context)
    {
        foreach (var matcherParameter in
        _matcher.Template.Parameters)
        {
            context.Values.Remove(matcherParameter.Name); //
            make sure none of the domain-placeholders are
            appended as query string parameters
        }

        return _innerRoute.GetVirtualPath(context);
    }

    private static void MergeValues(IDictionary<string, object>
    destination, IReadOnlyDictionary<string, object> values)
    {
        foreach (var kvp in values)
        {
            // This will replace the original value for the
            specified key.
            // Values from the matched route will take
            preference over previous
            // data in the route context.
            destination[kvp.Key] = kvp.Value;
        }
    }

    private void EnsureLoggers(HttpContext context)
    {
        if (_logger == null)
        {
            var factory =
            context.RequestServices.GetRequiredService<ILoggerFa
            ctory>();
            _logger = factory.Create<TemplateRoute>();
        }
    }

    public override string ToString()
    {
        return _domainTemplate + "/" + RouteTemplate;
    }
}
```

```
}
```
View DomainTemplateRoute - GetVirtualPath
DomainTemplateRouteBuilderExtensions.cs
Raw
File suppressed. Click to show.
```
1. 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
   26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
using System;
using System.Collections.Generic;
using Microsoft.AspNet.Routing;
using Microsoft.Framework.DependencyInjection;

namespace Multitenancy.Routing
{
    public static class DomainTemplateRouteBuilderExtensions
    {
        public static IRouteBuilder MapDomainRoute(this
        IRouteBuilder routeCollectionBuilder, string name, string
        domainTemplate, string routeTemplate)
        {
            MapDomainRoute(routeCollectionBuilder, name,
            domainTemplate, routeTemplate, (object)null);
            return routeCollectionBuilder;
        }

        public static IRouteBuilder MapDomainRoute(this
        IRouteBuilder routeCollectionBuilder, string name, string
        domainTemplate, string routeTemplate, object defaults, bool
        ignorePort = true)
        {
            return MapDomainRoute(routeCollectionBuilder, name,
            domainTemplate, routeTemplate, defaults, null,
            ignorePort);
        }

        public static IRouteBuilder MapDomainRoute(this
        IRouteBuilder routeCollectionBuilder, string name, string
        domainTemplate, string routeTemplate, object defaults,
        object constraints, bool ignorePort = true)
        {
            return MapDomainRoute(routeCollectionBuilder, name,
            domainTemplate, routeTemplate, defaults, constraints,
            null, ignorePort);
        }

        public static IRouteBuilder MapDomainRoute(this
        IRouteBuilder routeCollectionBuilder, string name, string
        domainTemplate, string routeTemplate, object defaults,
        object constraints, object dataTokens, bool ignorePort =
        true)
        {
            if (routeCollectionBuilder.DefaultHandler == null)
                throw new InvalidOperationException("Default handler
                must be set.");
```

```
            var inlineConstraintResolver =
            routeCollectionBuilder.ServiceProvider.GetRequiredServic
            e<IInlineConstraintResolver>();
            routeCollectionBuilder.Routes.Add(new
            DomainTemplateRoute(routeCollectionBuilder.DefaultHandle
            r, name, domainTemplate, routeTemplate,
            ObjectToDictionary(defaults),
            ObjectToDictionary(constraints),
            ObjectToDictionary(dataTokens), ignorePort,
            inlineConstraintResolver));
            return routeCollectionBuilder;
        }

        private static IDictionary<string, object>
        ObjectToDictionary(object value)
        {
            return value as IDictionary<string, object> ?? new
            RouteValueDictionary(value);
        }
    }
}
```

View DomainTemplateRoute - GetVirtualPath
ITenantFeature.cs
Raw
File suppressed. Click to show.

```
1. 2 3 4 5 6 7
namespace Multitenancy.Features
{
    public interface ITenantFeature
    {
        Tenant Tenant { get; }
    }
}
```

View DomainTemplateRoute - GetVirtualPath
Tenant.cs
Raw
File suppressed. Click to show.

```
1. 2 3 4 5 6 7
namespace Multitenancy.Features
{
    public class Tenant
    {
        public string Id { get; set; }
    }
}
```

View DomainTemplateRoute - GetVirtualPath
TenantFeature.cs
Raw
File suppressed. Click to show.

```
1. 2 3 4 5 6 7 8 9 10 11 12 13
namespace Multitenancy.Features
{
    public class TenantFeature
        : ITenantFeature
```

```
    {
        public TenantFeature(Tenant tenant)
        {
            Tenant = tenant;
        }

        public Tenant Tenant { get; private set; }
    }
}
```

View DomainTemplateRoute - GetVirtualPath
TenantResolverMiddleware.cs
Raw
File suppressed. Click to show.
1. 2