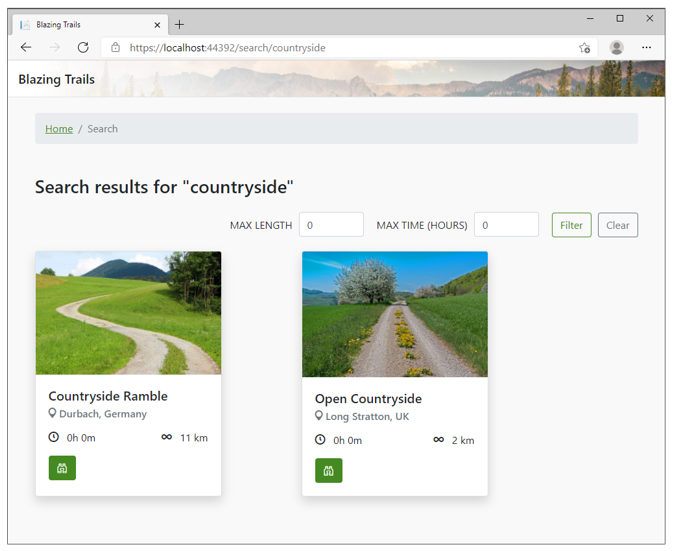
Yönlendirme veya navigasyon, web uygulamaları oluştururken temel bir kavramdır. Geleneksel olarak, bir sayfadan diğerine geçiş, sunucudan tamamen yeni, fiziksel bir HTML sayfası yükleme durumuydu. MVC veya Razor Pages gibi daha modern sunucu tabanlı çerçevelerde, bu sayfalar istemciye gönderilmeden önce sunucuda dinamik olarak derlenir, ancak süreç hala aynıdır. Ancak, tek sayfalı uygulamalarda işler biraz farklı çalışır.

Her zaman olduğu gibi, bu bölümde ortaya çıkaracağımız kavramları öğrenmemize yardımcı olması için Blazing Trails'e yeni bir özellik ekleyeceğiz. 

Bu sefer, uygulamaya bir arama işlevi ekleyeceğiz. Bu, kullanıcının ada veya konuma göre bir iz aramasına izin verecek. Arama terimini URL aracılığıyla ileterek, ana sayfadan arama sayfasına programlı olarak gitmek için yönlendirmeyi kullanacağız. Arama sayfasında bir kez onu ayıklayacağız ve ardından terimle eşleşen tüm izleri bulacağız.

Ayrıca herhangi bir sonucu filtreleme özelliğini de geliştireceğiz. Kullanıcının herhangi bir filtre de dahil olmak üzere aramayı işaretlemesine ve sonuçları doğrudan yeniden yüklemesine olanak tanıyan birden çok rota şablonu ve sorgu dizesi kullanarak bunu yapmanın birkaç yolunu keşfedeceğiz.

4.1 İstemci tarafı yönlendirmeye giriş

İstemci tarafı yönlendirme, sunucu tabanlı web uygulamalarında geleneksel gezinmeden önemli ölçüde farklıdır. Geleneksel çok sayfalı uygulamalarda, başka bir sayfaya gitmek için sunucuya yeni sayfa için bir istek yapılır. Yeni sayfa daha sonra tarayıcıya indirilir ve tarayıcı onu işler. SPA'larda, genel olarak konuşursak, tüm sayfalar istemcide bulunur ve bunlar arasında gezinme, istemci tarafı yönlendirici tarafından gerçekleştirilir.

4.1.1 Blazors yönlendirici

Blazor'da yönlendirici yalnızca başka bir bileşendir ve onu Uygulama bileşeninin (App.razor) içinde bulabilirsiniz. Liste 4.1, yönlendirici bileşenini gösterir.

Liste 4.1 App.razor dosyasında bulunan yönlendirici bileşeni

<Router AppAssembly="@typeof(Program).Assembly">

<Found Context="routeData">

<RouteView RouteData="@routeData"

DefaultLayout="@typeof(MainLayout)" />

<FocusOnNavigate RouteData="@routeData" Selector="h1" />

</Found>

<NotFound>

<PageTitle>Not found</PageTitle>

<LayoutView Layout="@typeof(MainLayout)">

<p>Sorry, there's nothing at this address.</p>

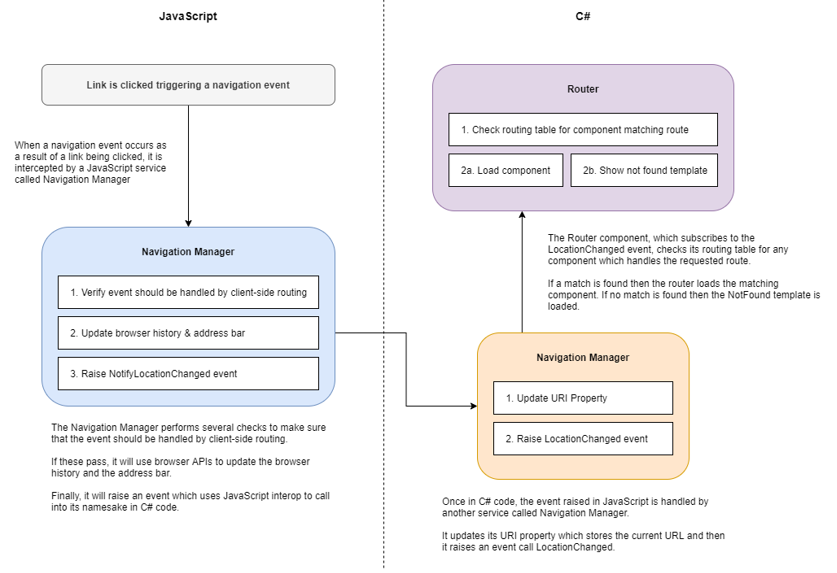
</LayoutView>

</NotFound>

</Router>

Bir Blazor uygulaması ilk yüklendiğinde, yönlendirici bileşeni, yönlendirilebilir bileşenleri veya benim sayfa bileşenleri olarak adlandırmayı tercih ettiğim şeyi bulmak için uygulamanın derlemelerini taramak *için yansıma* kullanır . Bunlar, *@page* olarak adlandırılan özel bir yönergeye sahip bileşenlerdir . AppAssemblyParametre üzerinden hangi montajın taranacağını bilir . Daha büyük uygulamalarda, sayfa bileşenlerinin birden çok derlemede olması mümkündür. Bu durumda yönlendirici bileşeni, AdditionalAssembliesbir IEnumerable<Assembly>.

Sayfa yönergesi, bileşenin hangi rota için yükleneceğini belirtmemize izin verir. Yönlendirici daha sonra bileşenin türünü ve işlediği rotayı bir yönlendirme tablosunda saklar. Yönlendirici daha sonra şekil 4.2'de açıklanan süreç aracılığıyla navigasyon olaylarını dinler.



Bir bağlantı tıklandığında, bu bir gezinme olayını tetikler. Blazor, JavaScript dünyasında yaşayan bir altyapıya sahiptir ve kodun yaptığı şeylerden biri, navigasyon olayları da dahil olmak üzere çeşitli olayları engellemektir.

*Bağlantının işaret ettiği URL, NavigationManager* adlı bir JavaScript hizmetine iletilir , bu hizmet, olayın istemci tarafı yönlendirmesi tarafından ele alınması gerektiğini doğrulamak için birkaç kontrol gerçekleştirir. Bu kontroller şunları içerir:

* Blazor'un yönlendiricisinin etkin olduğundan emin olmak
* Kullanıcının bağlantıyı yeni bir sekmede/pencerede açmak istediğini bildiren shift veya kontrol gibi hiçbir değiştirici tuşa basılmamış olması
* Bağlantıda hiçbir hedef özelliğinin bulunmadığını. Yine, bu, bağlantının başka bir sekmede/pencerede açılması gerektiğinin sinyalini verir. Bunun tek istisnası, \_self hedefidir, bu, geçerli sekmede/pencerede aç anlamına gelir.
* Bağlantı, Blazor uygulamasının ana bilgisayar sayfasında tanımlanan temel etiket kapsamına girmelidir.

Tüm bu kontroller başarılı olursa, tarayıcı geçmişi URI ile birlikte güncellenecektir. Bu, tarayıcı ileri ve geri düğmesi gibi özelliklerin çalışmasını ve geleneksel tam sayfa gezinme görünümünü sağlar. Hizmetin attığı son adım, bazı JavaScript birlikte çalışmalarını tetikleyen bir olay oluşturmaktır. Bu olay, C# dünyasında yaşayan *NavigationManager* adlı aynı adlı bir hizmet tarafından alınır .

C# NavigationManager olayı aldığında, URI özelliğini günceller. Bu özellik, bileşenlerin gerektiğinde erişebilmesi için geçerli URL'yi saklar. Daha sonra *LocationChanged* adlı bir olayı tetikler . Blazor'un yönlendiricisi bu olaya abone olur ve tetiklendiğinde yönlendirici bir eşleşme bulmak için *NavigationManager'ın URI özelliğini yönlendirme tablosuna göre kontrol eder.*Bir tane bulunursa, bileşen yüklenir, aksi halde Bulunamadı şablonu oluşturulur.

**4.1.2 Sayfa bileşenlerini tanımlama**

*Sayfa bileşenleri, belirli bir yönerge bildiren düzenli bileşenlerdir, bu yönergeye @page yönergesi* denir . Yönerge adı ve rota şablonu olmak üzere iki bölümü vardır ve bildirildiğinde şöyle görünür .

@page "/my-awesome-page"

Rota şablonu tırnak içindeki bölümdür, bu, bileşenin işleyeceği URL'yi tanımlar – her zaman " /"karakteri ile başlamalıdır, aksi takdirde bir derleyici hatası alırsınız. Ayrıca, rota şablonları benzersiz olmalıdır. Aynı şeyi bildirmemeye dikkat edin. Bu, bir çalışma zamanı hatasıyla sonuçlanacağından, birden çok bileşende route şablonu.Daha sonra göreceğimiz gibi, tek bir bileşenin birden çok @page yönergesi bildirmesi ve birden çok yolu işlemesi gayet iyi.

Blazing Trails için yeni Arama özelliğimizi oluşturmaya başlamak için uygulamaya yeni bir sayfa bileşeni ekleyelim. Özellikler > Ana klasöre SearchPage.razor adlı liste 4.2'de gösterilen kodla yeni bir jilet bileşeni ekleyin.

@page "/search"

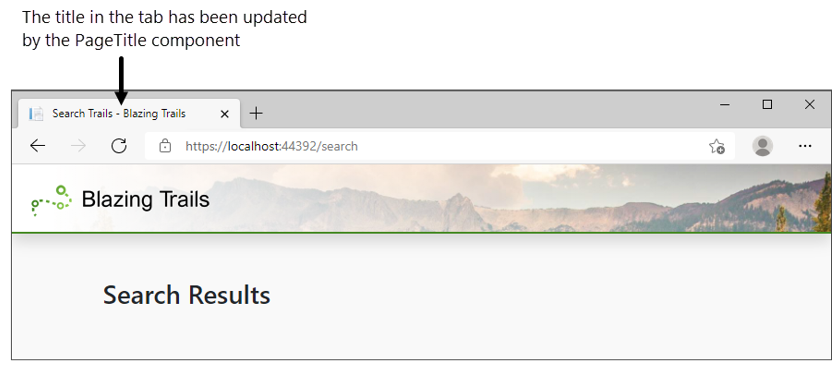
<PageTitle>Search Trails - Blazing Trails</PageTitle>

<h3>Search results</h3>

4.2 listesinden de görebileceğiniz gibi, sayfa bileşenlerini tanımlamak oldukça basittir. @page yönergesi sayfanın en üstüne eklenir ve ardından bileşenin işlemesi gereken rota, rota şablonu kullanılarak tırnak içinde belirtilir.

Daha önce bahsettiğim gibi, rota şablonu tırnak içinde olmalı ve bir /. Bu eksikse, derleyiciden bir derleme hatası alırsınız.

Artık uygulamayı oluşturabilir ve çalıştırabilirsiniz; Henüz bağlantı vermediğimiz için arama sayfasının adresini manuel olarak yazmanız gerekecek. Şekil 4.3 yeni sayfanın nasıl göründüğünü gösterir.



Bu şu anda çok ilham verici değil ama bir sonraki bölümde bu sayfaya bağlantı verecek arama kutusunu oluşturacağız.

**4.2 Programlı olarak sayfalar arasında gezinme**

Web sayfaları oluştururken öğrendiğimiz ilk şeylerden biri, köprüler kullanarak bir sayfayı diğerine nasıl bağlayacağımızdır. Bu yöntem hala modern SPA uygulamalarının temelini oluşturuyor ve tam olarak geleneksel sunucu tabanlı bir uygulamada olduğu gibi çalışıyor, ancak programlı navigasyonun gerekli olduğu birçok senaryo var.

Blazor'da programlı gezinme bu NavigationManager.NavigateTo()yöntemle sağlanır. Bunu, arama terimlerini girdikten sonra kullanıcıları yeniden yönlendirmek için kullanacağız. Ama önce, Ana özellik klasöründe TrailSearch.razor adlı yeni bir bileşen oluşturmamız gerekiyor. Bu bileşen, arama kutusuna ve Arama Sayfasına yeniden yönlendirme mantığına ev sahipliği yapacaktır – liste 4.3 kodu gösterir.