Phần 10: Thêm tìm kiếm

Bài này sẽ hướng dẫn bạn cách thêm mục tìm kiếm vào giao diện trang Index.

# Thêm phương thức Search và View tương ứng

Ở giao diện trang Index (http://localhost:xxxx/Link), bạn có thể thấy thiếu một ô tìm kiếm theo tên liên kết (LinkName), mô tả liên kết (LinkDescription), hay bất cứ trường thuộc tính nào khác. Bây giờ, chúng ta sẽ tiến hành làm điều này.

# Cập nhật phương thức Index

Đầu tiên, bạn mở mã nguồn tập tin Controllers/LinkController.cs và sửa phương thức Index thành như sau:

// GET: /Link/

public ActionResult Index(string searchString)

{

var links = from l in db.Links // lấy toàn bộ liên kết

select l;

if (!String.IsNullOrEmpty(searchString)) // kiểm tra chuỗi tìm kiếm có rỗng/null hay không

{

links = links.Where(s => s.LinkName.Contains(searchString)); //lọc theo chuỗi tìm kiếm

}

return View(links); //trả về kết quả

}

Đoạn mã vừa thay đổi ở phương thức Index hoạt động rất đơn giản như sau. Biến **searchString** kiểu string là chuỗi tìm kiếm cần nhập vào. Đầu tiên, tạo 1 biến **var links** để làm câu truy vấn lấy tất cả Link từ database, lưu ý kỹ bước này chưa kết nối với database mà chỉ là câu truy vấn. Mệnh đề if kiểm tra nếu biến searchString không null hoặc rỗng (empty) thì thêm điều kiện lọc vào biến **links**, còn không thì để như ban đầu. Sau đó trả về kết quả, đến đây câu truy vấn mới được hình thành và tương tác với database.

Lấy ví dụ thực tế, nếu bạn có biến searchString có giá trị “Dammio” thì chuỗi truy vấn SQL trong biến links được biên dịch từ [LINQ](https://www.dammio.com/glossary/linq) sẽ là: “**select \* from Links where LinkName like ‘Dammio’**“. Còn nếu biến searchString không có giá trị thì chuỗi truy vấn sẽ là: “**select \* from Links**“. Cách hoạt động rất đơn giản phải không nào?

Giải thích thêm về đoạn

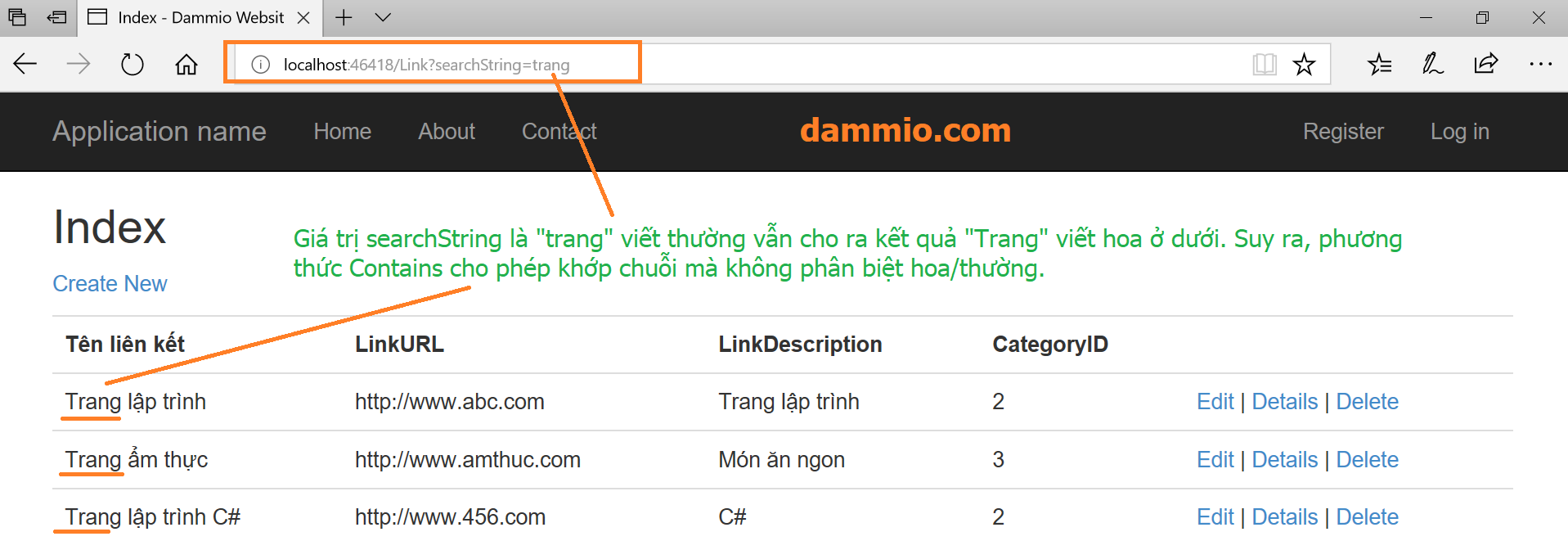
links.Where(s => s.LinkName.Contains(searchString)),

mệnh đề Where thì tương tự như mệnh đề where trong ngôn ngữ truy vấn SQL. Phương thức Contains có nghĩa bao hàm, chứa trong. Như vậy, chúng ta tìm các link sao cho có chứa cụm từ là giá trị của biến searchString. OK, tiếp theo bạn thấy cú pháp s => s.LinkName, đây chính là diễn giải Lambda. Tại sao lại dùng Lambda? Lambda được dùng trong các truy vấn [LINQ](https://www.dammio.com/glossary/linq) dựa trên phương thức (như phương thức Where) như là cách đối số để truy vấn các phương thức toán tử để mục đích biểu diễn gọn nhẹ và rút gọn mã nguồn. Nếu bạn còn lăn tăn về Lambda thì hãy xem bài [Giới thiệu về LINQ](https://www.dammio.com/2016/12/03/phan-1-linq-gioi-thieu-ve-linq) để nắm vững.

Sau khi thêm xong, build dự án (Ctrl + Shift + B), và bạn thử chạy liên kết **http://localhost:xxxx/Link** trở trình duyệt, mình có thêm 1 số dữ liệu liên kết để demo như sau.



Rồi, sau đó thử chạy lại đường dẫn thêm đoạn truy vấn **http://localhost:46418/Link?searchString=trang** như sau.



Bạn có thấy, trong [URL](https://www.dammio.com/glossary/url) trên, tôi cố ý để chữ “trang” viết thường mà kết quả vẫn tìm ra các liên kết có tên chữ “Trang” viết hoa chữ T. Độc đáo phải không, suy ra phương thức Contains trong đoạn mã Index cho phép so khớp hoa/thường. Vì vậy, chức năng tìm kiếm với ASP.NET [MVC](https://www.dammio.com/glossary/mvc) đơn giản và khỏe hơn bao giờ hết.

# Sửa đường dẫn URL

Nhược điểm của http://localhost:xxxx/Link?searchString=trang là bạn phải thêm đuôi truy vấn ?searchString=trang vào cuối [URL](https://www.dammio.com/glossary/url). Điều đó khiến đường dẫn trở nên rườm ra theo cách viết Web cũ và giảm SEO. Bạn mong muốn có đường dẫn dạng như http://localhost:xxxx/Link/trang thì sẽ tìm kiếm các liên kết có chữ “trang”? Nếu vậy, bạn vào tập tin **App\_Start\RouteConfig.cs** để sửa. Tuy nhiên, vì khi bạn tập tin này thì đường dẫn toàn bộ dự án web có thể bị ảnh hưởng, cách tốt nhất là bạn sửa đoạn mã ở Index theo như đường dẫn mặc định ở **App\_Start\RouteConfig.cs** đó là [**url**](https://www.dammio.com/glossary/url)**: “{controller}/{action}/{id}”**. Khi bạn rành ASP.NET MC hơn, bạn có thể sửa tùy theo ý muốn.

Bạn mở Controllers/LinkController.cs và sửa phương thức Index như sau.

public ActionResult Index(string id)

{

var links = from l in db.Links

select l;

if (!String.IsNullOrEmpty(id))

{

links = links.Where(s => s.LinkName.Contains(id));

}

return View(links);

}

Đến đây, build dự án (Ctrl + Shift + B), chạy đường dẫn http://localhost:46418/Link/Index/trang để xem kết quả. Nếu bạn muốn một đường dẫn dạng như **http://localhost:46418/Link/trang** thì bạn lại phải cấu hình ở **App\_Start\RouteConfig.cs**, tạm thời chúng ta để vấn đề này sau.

# Tạo giao diện tìm kiếm ở View

Bạn không thể lúc nào search cụm từ bằng cách chạy đường dẫn được, vì vậy, bạn phải tạo ô tìm kiếm ở View. Do đó, ở tập tin **Controllers/LinkController.cs** phải thay đổi phương thức Index về cách cũ như sau:

// GET: /Link/

       public ActionResult Index(string searchString)

       {

           var links = from l in db.Links

                       select l;

           if (!String.IsNullOrEmpty(searchString))

           {

               links = links.Where(s => s.LinkName.Contains(searchString));

           }

           return View(links);

       }

Tiếp theo, bạn mở tập tin **Views\Link\Index.cshtml** và sau đoạn **@Html.ActionLink(“Create New”, “Create”)** bạn thêm đoạn mã:

@model IEnumerable<DammioMVC.Models.Link>

@{

ViewBag.Title = "Index";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h2>Index</h2>

<p>

@Html.ActionLink("Create New", "Create")

<!-- Đoạn cần thêm -->

@using (Html.BeginForm())

{

<p>

Title: @Html.TextBox("SearchString") <input type="submit" value="Tìm kiếm" />

</p>

}

<!-- Kết thúc -->

</p>

Ở đoạn mã trên, chúng ta dùng Html.BeginForm() để tạo 1 phần tử form mặc định dùng **HttpPost**, với ô tìm kiếm là 1 textbox (@Html.TextBox) và nút nhấn Submit. Đến đây, hãy build dự án (Ctrl + Shift + B) bạn chạy đường dẫn **http://localhost:xxxx/Link** và nhập vào ô tìm kiếm giá trị “Lập trình” thì thu được kết quả.



Bạn có thể sử dụng kiểu GET (HttpGet) bằng cách thay đổi Html.BeginForm ở tập tin **Views/Link/Index.cshtml** như sau.

<h2>Index</h2>

<p>

    @Html.ActionLink("Create New", "Create")

    <!-- Đoạn cần thêm -->

    @using (Html.BeginForm("Index","Link",FormMethod.Get)) // -- phần thay đổi

    {

    <p>

        Title: @Html.TextBox("SearchString") <input type="submit" value="Tìm kiếm" />

    </p>

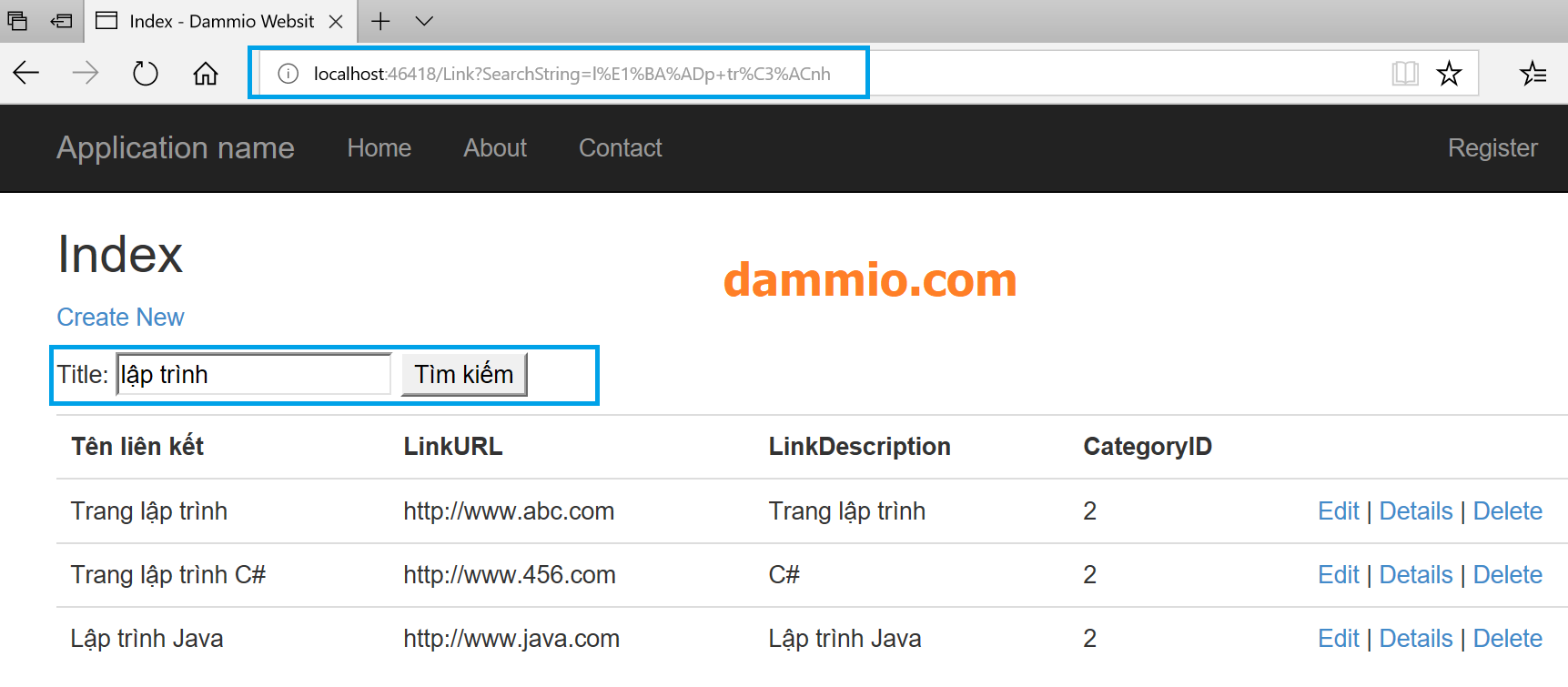
    }

    <!-- Kết thúc -->

</p>

Trong đoạn mã trên, phương thức Html.BeginForm(“Index”,”Link”,FormMethod.Get) chứa 3 đối số, “Index” là phương thức Index(), “Link” là tên Controller và FormMethod.Get là định nghĩa form kiểu POST hay GET.

Sau đó, bạn chạy link **http://localhost:xxxx/Link** thử gõ vào ô tìm kiếm “Lập trình” và nhấn nút Tìm kiếm để xem kết quả.

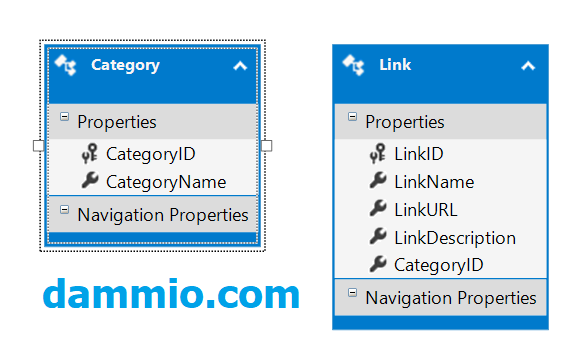


# Tìm kiếm theo thể loại (Category)

Như bạn để ý thiết kế database từ đầu, bảng Link có liên kết với bảng Category theo mối quan hệ 1-nhiều. Một Link chỉ thuộc về 1 Category duy nhất. Đến đây chúng ta cũng tạo Controller cho bảng Category tương tự các bài trước. Bạn cũng truy cập **http://localhost:xxxx/Category/Index** để tạo 1 số danh mục như sau.



Tiếp theo, chúng ta bật tập tin **Model/Model.edmx** để xem mô hình database như sau.



Theo hình trên, bạn có thể thấy chúng ta không có tạo bất cứ ràng buộc nào nữa Category và Link thông qua **các thuộc tính điều hướng (navigation links)**. Vì vậy, để lấy dữ liệu kết hợp giữa 2 bảng Link và Category là một cách không dễ dàng. Tuy nhiên, **trong bài viết tôi sẽ chỉ cách lấy này để cho các bạn biết trên thực tế đôi khi bạn gặp trường hợp “oái ăm” như thế này.** Lưu ý cách dễ hơn là khi bạn tạo liên kết giữa các bảng thì bạn có thể thấy các thuộc tính dạng như public virtual Category Category {get, set}… và có thể dùng phương thức Include trong mệnh đề truy vấn để lấy dữ liệu. Bạn có thể đọc thêm bài viết [Reading Related Data with the Entity Framework in an ASP.NET MVC Application](https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/getting-started-with-ef-using-mvc/reading-related-data-with-the-entity-framework-in-an-asp-net-mvc-application) để hiểu rõ cách lấy.

Đầu tiên, bạn vào tập tin Model/Link.cs thêm dòng **public string CategoryName { get; set; }** vào như sau.

namespace DammioMVC.Models

{

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations; // thêm bằng tay

public partial class Link

{

public int LinkID { get; set; }

[Display(Name = "Tên liên kết")] // thêm bằng tay

[Required]

public string LinkName { get; set; }

public string LinkURL { get; set; }

public string LinkDescription { get; set; }

public Nullable<int> CategoryID { get; set; }

public string CategoryName { get; set; } // thêm ở đây

}

}

Ok, sau đó quay trở lại tập tin **Controllers/LinkController.cs**để sửa phương thức Index như sau.

public ActionResult Index(string searchString, int categoryID = 0)

{

//1. Tạo danh sách danh mục để hiển thị ở giao diện View thông qua DropDownList

var categories = from c in db.Categories select c;

ViewBag.categoryID = new SelectList(categories, "CategoryID", "CategoryName"); // danh sách Category

//2. Tạo câu truy vấn kết 2 bảng Link, Category bằng mệnh đề join

var links = from l in db.Links

join c in db.Categories on l.CategoryID equals c.CategoryID

select new { l.LinkID, l.LinkName, l.LinkURL, l.LinkDescription, l.CategoryID, c.CategoryName };

//3. Tìm kiếm chuỗi truy vấn

if (!String.IsNullOrEmpty(searchString))

{

links = links.Where(s => s.LinkName.Contains(searchString));

}

//4. Tìm kiếm theo CategoryID

if (categoryID != 0)

{

links = links.Where(x => x.CategoryID == categoryID);

}

//5. Chuyển đổi kết quả từ var sang danh sách List<Link>

List<Link> listLinks = new List<Link>();

foreach (var item in links)

{

Link temp = new Link();

temp.CategoryID = item.CategoryID;

temp.CategoryName = item.CategoryName;

temp.LinkDescription = item.LinkDescription;

temp.LinkID = item.LinkID;

temp.LinkName = item.LinkName;

temp.LinkURL = item.LinkURL;

listLinks.Add(temp);

}

return View(listLinks);

}

Trong đoạn mã trên, bạn tạo thêm 1 biến categoryID để tìm kiếm theo danh mục. Sau đó ở (1) là tạo danh sách các danh mục, hiển thị danh sách này ở DropDownList bên View Index.cshtml. Tiếp, bạn tạo câu truy vấn (2) để kết 2 bảng Link, Category. Mục (3) và (4) kiểm tra dữ liệu, nếu có đầu vào là từ khóa cần tìm và ID danh mục thì thêm vào điều kiện lọc để tìm, không thì thôi. Mục (5) chuyển dữ liệu từ kiểu không xác định var sang danh sách List.

Ở tập tin **Views/Link/Index.cshtml**, bạn thay đổi nội dung mã nguồn như sau.

@model IEnumerable<DammioMVC.Models.Link>

@{

ViewBag.Title = "Index";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h2>Index</h2>

<p>

@Html.ActionLink("Create New", "Create")

<!-- 1. Đoạn cần thêm -->

@using (Html.BeginForm("Index","Link",FormMethod.Get))

{

<p>

Title: @Html.TextBox("SearchString") --- Category: @Html.DropDownList("categoryID", "All") <input type="submit" value="Tìm kiếm" />

</p>

}

<!-- Kết thúc -->

</p>

<table class="table">

<tr>

<th>

@Html.DisplayNameFor(model => model.LinkName)

</th>

<th>

@Html.DisplayNameFor(model => model.LinkURL)

</th>

<th>

@Html.DisplayNameFor(model => model.LinkDescription)

</th>

<th>

@Html.DisplayNameFor(model => model.CategoryID)

</th>

<!-- 2. Đoạn cần thêm-->

<th>

@Html.DisplayNameFor(model => model.CategoryName)

</th>

<!-- Kết thúc -->

<th></th>

</tr>

@foreach (var item in Model) {

<tr>

<td>

@Html.DisplayFor(modelItem => item.LinkName)

</td>

<td>

@Html.DisplayFor(modelItem => item.LinkURL)

</td>

<td>

@Html.DisplayFor(modelItem => item.LinkDescription)

</td>

<td>

@Html.DisplayFor(modelItem => item.CategoryID)

</td>

<!-- 3. Đoạn cần thêm-->

<td>

@Html.DisplayFor(modelItem => item.CategoryName)

</td>

<!-- Kết thúc -->

<td>

@Html.ActionLink("Edit", "Edit", new { id=item.LinkID }) |

@Html.ActionLink("Details", "Details", new { id = item.LinkID }) |

@Html.ActionLink("Delete", "Delete", new { id = item.LinkID })

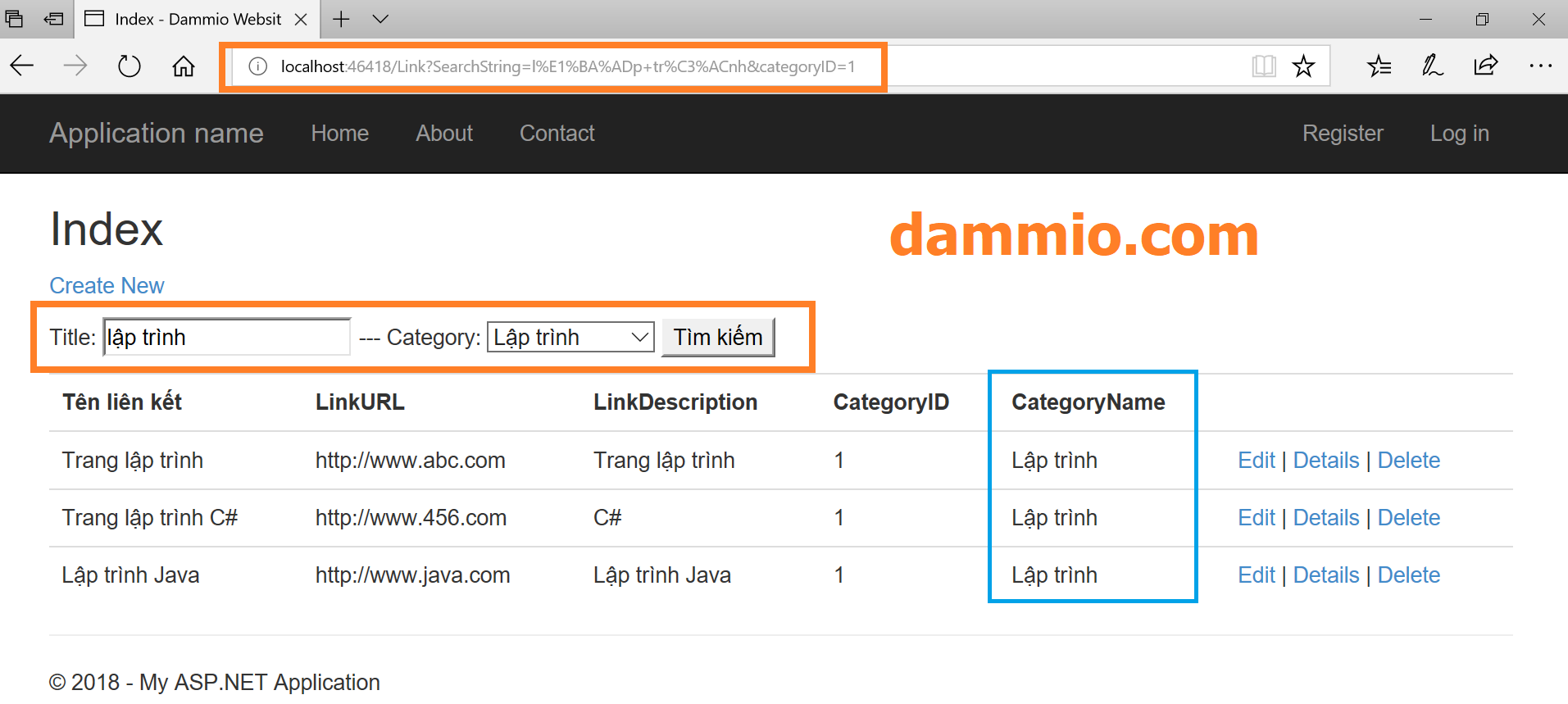
</td>

</tr>

}

</table>

Trong đoạn mã trên, bạn thêm (1) để tạo form kèm theo danh sách lựa chọn DropDownList, lấy biến từ ViewBag.categoryID. Ở (2) và (3) chúng ta thêm một trường CategoryName để hiển thị tên danh mục của từng liên kết.



1. **Kết luận**

Trong bài, bạn đã tìm hiểu cách thêm tìm kiếm dữ liệu trong [MVC](https://www.dammio.com/glossary/mvc) thông qua các ví dụ cơ bản. Có nhiều cách tìm kiếm khác mà bạn cần phải tham gia xây dựng nhiều dự án ASP.NET [MVC](https://www.dammio.com/glossary/mvc) mới có kinh nghiệm và kiến thức.

[1]. [DammioMVC\_Phan10.rar](https://www.dammio.com/wp-content/uploads/2018/07/DammioMVC_Phan10.rar)