Phần 12: Thêm sắp xếp

Sau khi thêm phần tìm kiếm vào trang **Link/Index.cshtml** để tìm kiếm liên kết (Link) theo từ khóa và danh mục (Category). Bạn có thể học cách thêm sắp xếp (order) theo các cột hiển thị trong ASP.NET [MVC](https://www.dammio.com/glossary/mvc) trong bài này. Bài này sẽ sử dụng lại dự án ở phần 10 hoặc phần 11 để tiếp tục thêm mã nguồn tạo sắp xếp nhé.

# Thêm sắp xếp

Trong bài này, hai tập tin **LinkController.cs và Views/Link/Index.cshtml** đã có chứa mã tìm kiếm ở phần 10 và phần 11 sẽ bị sửa lại, bạn chỉ cần làm theo hướng dẫn bài này.

# Sửa tập tin Controller/LinkController.cs

Để sắp xếp tăng hay giảm, trước hết bạn cần 1 biến tên là “**sortOrder**” (hay đặt bất cứ tên nào bạn muốn) để lưu trữ trạng thái sắp xếp tăng, giảm trên trường thuộc tính nào trong bảng Link. Sau đó, bạn cần 1 biến ViewBag để giữ trạng thái trên Views (giao diện) tên là NameSortParm.

public ActionResult Index(string sortOrder)

{

// 1. Thêm biến NameSortParm để biết trạng thái sắp xếp tăng, giảm ở View

ViewBag.NameSortParm = String.IsNullOrEmpty(sortOrder) ? "name\_desc" : "";

// 2. Truy vấn lấy tất cả đường dẫn

var links = from l in db.Links

select l;

// 3. Thứ tự sắp xếp theo thuộc tính LinkName

switch (sortOrder)

{

// 3.1 Nếu biến sortOrder sắp giảm thì sắp giảm theo LinkName

case "name\_desc":

links = links.OrderByDescending(s => s.LinkName);

break;

// 3.2 Mặc định thì sẽ sắp tăng

default:

links = links.OrderBy(s => s.LinkName);

break;

}

// 4. Trả kết quả về Views

return View(links.ToList());

}

Cách hoạt động đoạn mã trên như sau.

1. Đầu tiên, bạn truyền biến sortOrder vào phương thức Index, dòng code   
ViewBag.NameSortParm = String.IsNullOrEmpty(sortOrder) ? “name\_desc” : “”;   
dùng để kiểm tra giá trị của biến sortOrder, nếu biến này có giá trị rỗng thì đặt giá trị biến NameSortParm là “name\_desc”, ngược lại là rỗng. Cách viết 1 dòng là nguyên tắc tam phân (toán tử ?) dùng để tinh giản mã nguồn. Nếu viết dài dòng thì sẽ như sau:

// Bạn nào code "gà" sẽ viết như sau, còn PRO thì như trên (1 dòng duy nhất)

 if(String.IsNullOrEmpty(sortOrder))

{

      ViewBag.NameSortParm = "name\_desc";

}

else

{

      ViewBag.NameSortParm = "";

}

* 2. Tạo 1 câu truy vấn **var links = from l in db.Links select l;** để lấy hết tất cả liên kết trong bảng.
* 3. Bước này chúng ta dùng một mệnh đề switch … case (tối giản hơn so với if else) để lọc mệnh đề truy vấn theo sắp tăng hay giảm dựa theo trạng thái biến đầu vào sortOrder.
* 4. Sau đó, chúng ta trả về kết quả truy vấn trên giao diện để hiển thị.

# Sửa nội dung Views/Link/Index.cshtml

Ở tập tin View Index.cshtml bạn sửa như sau, chú ý đoạn cần thêm.

@model IEnumerable<DammioMVC1.Models.Link>

@{

ViewBag.Title = "Index";

}

<h2>Index</h2>

<p>

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

</p>

<table class="table">

<tr>

<!-- Đoạn cần thêm -->

<th>

@Html.ActionLink("Link Name", "Index", new { sortOrder = ViewBag.NameSortParm })

</th>

<!-- Kết thúc -->

<th>

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

</th>

<th>

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

</th>

<th>

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

</th>

<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.Category.CategoryName)

</td>

<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>

}

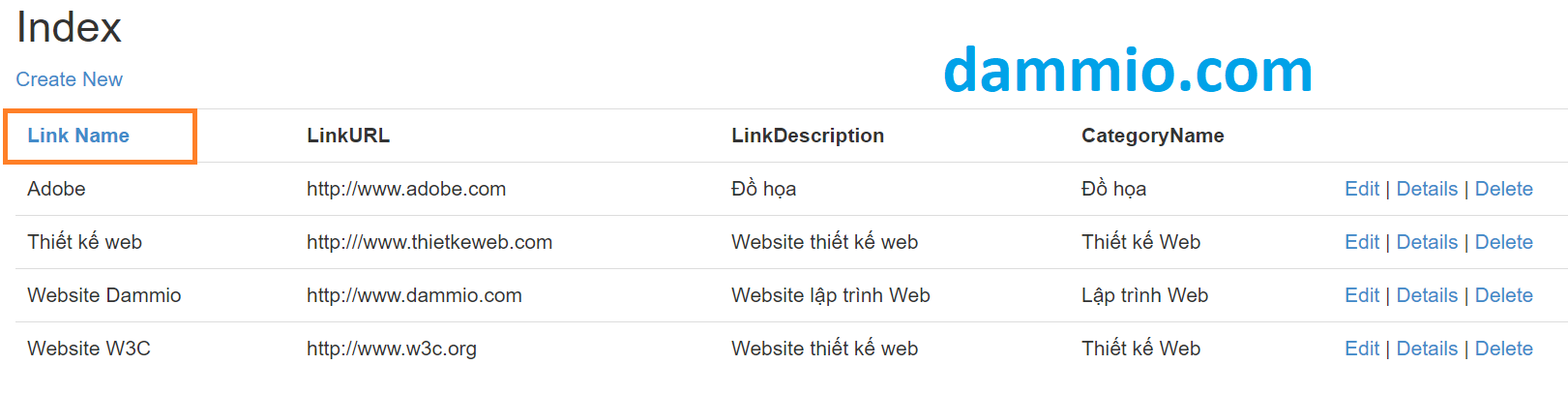
</table>

Bạn có thể thấy đoạn code trên, chúng ta chỉ thêm dòng mã **@Html.ActionLink(“Link Name”, “Index”, new { sortOrder = ViewBag.NameSortParm })** dùng để tạo một liên kết (**a href**) để người dùng có thể nhấn vào để sắp xếp theo trường LinkName tăng hay giảm.

Đoạn @Html.ActionLink(“Link Name”, “Index”, new { sortOrder = ViewBag.NameSortParm }) có thể diễn dịch thành mã HTML như sau:

<a href="/Link?sortOrder=giá trị của biến">Link Name</a>

Sau đó, bạn thử chạy trình duyệt http://localhost:xxxx/Link để có thể xem kết quả như sau. Bạn có thể nhấp lên liên kết LinkName (màu xanh) để xem kết quả sắp xếp.



Khi bạn sắp giảm theo tên (Z->A) thì sẽ có đường dẫn **http://localhost:xxxx/Link?sortOrder=name\_desc**, ngược lại sắp mặc định (tức A->Z) thì sẽ có đường dẫn **http://localhost:45033/Link**.

# Sắp xếp theo nhiều thuộc tính

Ở phần trước, bạn đã tìm cách sắp xếp tăng/giảm theo **thuộc tính LinkName**. Trên thực tế, các ứng dụng Web luôn đòi hỏi sắp xếp nhiều thuộc tính hơn, do đó phần này sẽ chỉ bạn cách sắp xếp theo nhiều thuộc tính.

Để sắp xếp thứ tự theo thuộc tính LinkDescription bên cạnh LinkName đã có, bạn làm theo hướng dẫn sau.

## Sửa tập tin Controller/LinkController.cs

Bạn mở tập tin LinkController.cs, tìm phương thức Index và sửa như sau.

public ActionResult Index(string sortOrder)

{

// 1. Thêm biến NameSortParm để biết trạng thái sắp xếp tăng, giảm ở View

ViewBag.NameSortParm = String.IsNullOrEmpty(sortOrder) ? "name\_desc" : "";

ViewBag.DescriptionSortParm = sortOrder == "Description" ? "description\_desc" : "description";

// 2. Truy vấn lấy tất cả đường dẫn

var links = from l in db.Links

select l;

// 3. Thứ tự sắp xếp theo thuộc tính LinkName

switch (sortOrder)

{

// 3.1 Nếu biến sortOrder sắp giảm thì sắp giảm theo LinkName

case "name\_desc":

links = links.OrderByDescending(s => s.LinkName);

break;

// 3.2 Sắp tăng dần theo LinkDescription

case "Description":

links = links.OrderBy(s => s.LinkDescription);

break;

// 3.4 Sắp giảm theo LinkDescription

case "description\_desc":

links = links.OrderByDescending(s => s.LinkDescription);

break;

// 3.5 Mặc định thì sẽ sắp tăng

default:

links = links.OrderBy(s => s.LinkName);

break;

}

// 4. Trả kết quả về Views

return View(links.ToList());

}

Trong đoạn mã trên, bạn chú ý chúng ta thêm 1 biến ViewBag **DescriptionSortParm** dùng để giữ trạng thái giá trị trên Views để biết khi nào sắp tăng hay giảm theo thuộc tính LinkDescription. Ở mệnh đề switch … case chúng ta cần thêm 2 trường hợp, **nếu giá trị của sortOrder là “Description” thì chúng ta sẽ sắp tăng theo LinkDescription, còn giá trị của sortOrder là “description\_desc” thì sẽ sắp giảm**.

## Sửa nội dung Views/Link/Index.cshtml

Trên tập tin View, bạn chỉ cần sửa dòng **@Html.ActionLink(“Link Description”, “Index”, new { sortOrder = ViewBag.DescriptionSortParm })** để tạo ra một liên kết (a href) mà người sử dụng có thể nhấp chuột để sắp xếp trên giao diện.

@model IEnumerable<DammioMVC1.Models.Link>

@{

    ViewBag.Title = "Index";

}

<h2>Index</h2>

<p>

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

</p>

<table class="table">

    <tr>

        <!-- Đoạn cần thêm -->

        <th>

            @Html.ActionLink("Link Name", "Index", new { sortOrder = ViewBag.NameSortParm })

        </th>

        <!-- Kết thúc -->

        <th>

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

        </th>

        <!-- Đoạn cần thêm -->

        <th>

            @Html.ActionLink("Link Description", "Index", new { sortOrder = ViewBag.DescriptionSortParm })

        </th>

        <!-- Kết thúc -->

        <th>

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

        </th>

        <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.Category.CategoryName)

        </td>

        <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 })

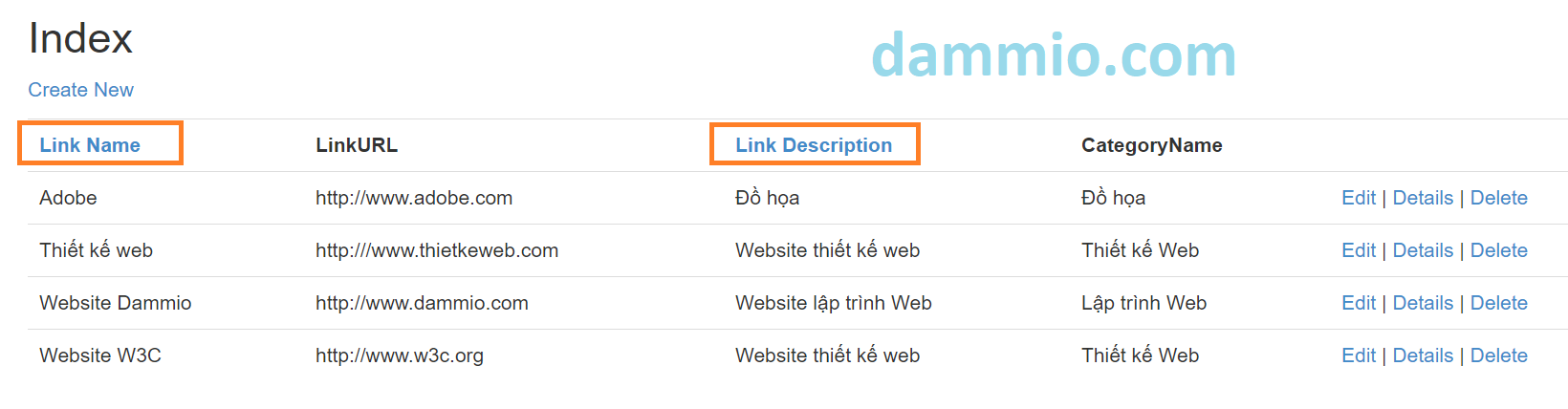
        </td>

    </tr>

}

</table>

Sau đó, bạn build dự án và chạy đường dẫn **http://localhost:xxxx/Link** để xem kết quả với giao diện như sau.



Sau đây là bảng ghi lại giá trị của các liên kết LinkName và LinkDescription theo thứ tự sắp xếp.

|  |  |  |
| --- | --- | --- |
| **Thứ tự sắp xếp hiện tại** | **Liên kết LinkName** | **Liên kết LinkDescription** |
| LinkName ascending | descending | ascending |
| LinkName descending | ascending | descending |
| LinkDescription ascending | ascending | descending |
| LinkDescription descending | ascending | ascending |

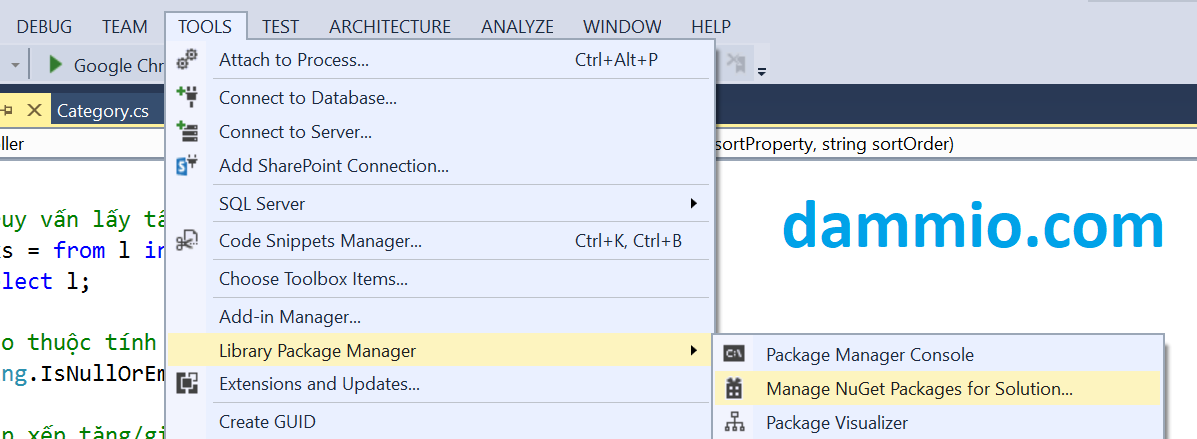
# Sắp xếp theo nhiều thuộc tính (nâng cao)

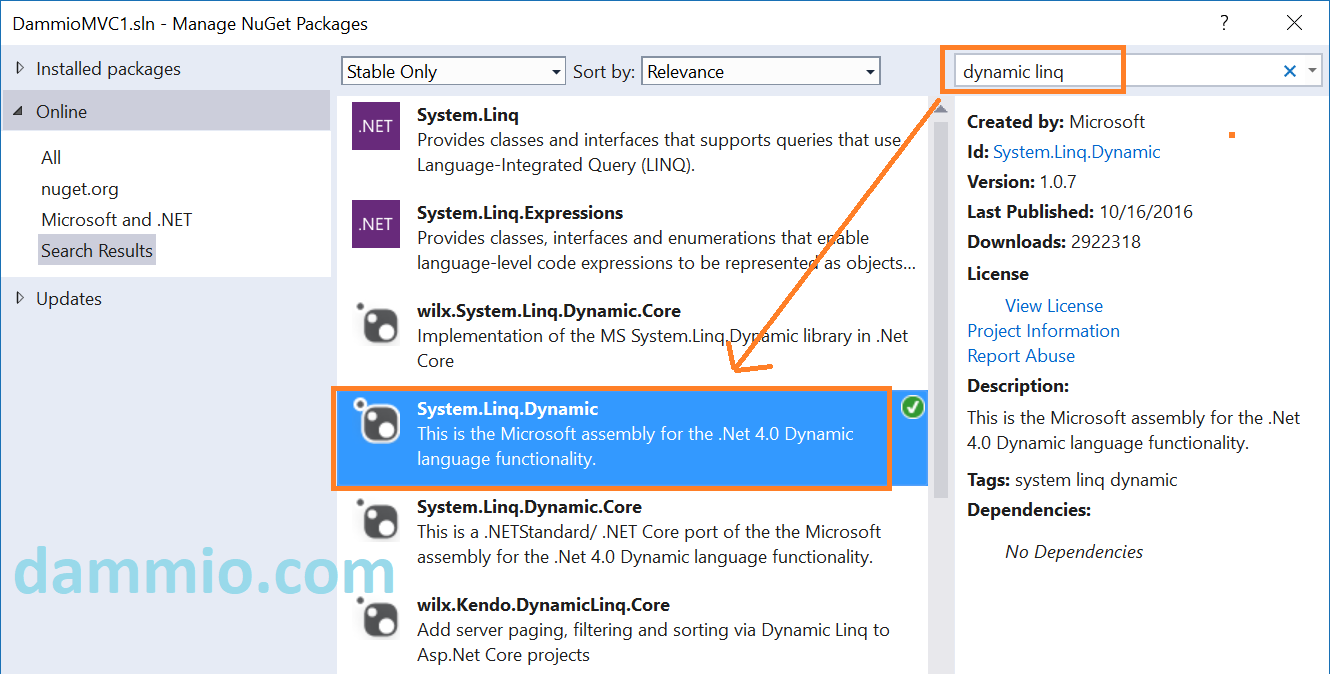
Đến đây, nhiều bạn sẽ tự hỏi, nếu sắp xếp thứ tự với 10 hay 100 thuộc tính thì chúng ta cũng sẽ làm như cách hướng dẫn trên? Nếu vậy, code sẽ dài dòng? Bạn giải quyết vấn đề này bằng cách dùng các control có trả phí rất cao như **Telerik, Syncfusion hay Dev Express**.

Nếu bạn không thể trả chi phí cho các website cung cấp control, bạn vẫn có thể code tay với cách na ná gần giống như trên. Mình sẽ chỉ bạn cách chỉ có ở website [**DAMMIO**](https://www.dammio.com/2018/10/26/asp-net-mvc-phan-12-them-sap-xep) như sau.

## Cài đặt Dynamic LINQ 1.0.7

Chúng ta phải sử dụng thư viện này để tạo 1 số câu truy vấn sql theo kiểu cũ thông qua 1 số hàm hỗ trợ. Trong Visual Studio, bạn vào Tools -> Library Package Manager -> Manage NuGet Packages…



Sau đó gõ **Dynamic [Linq](https://www.dammio.com/glossary/linq" \t "_blank)** bên ô Search góc trái trên cùng, chọn gói **System.**[**Linq**](https://www.dammio.com/glossary/linq)**.Dynamic** để cài đặt.  


Vậy là xong nhé!

## Sửa tập tin Controller/LinkController.cs

Bạn sửa phương thức Index của tập tin Controller/LinkController.cs này như sau.

using System.Linq.Dynamic; // nhúng vào tập tin

using System.Linq.Expressions; // nhúng vào tập tin

...

public ActionResult Index(string sortProperty, string sortOrder)

{

    // 1. Tạo biến ViewBag SortOrder để giữ trạng thái sắp tăng hay giảm

    ViewBag.SortOrder = String.IsNullOrEmpty(sortOrder) ? "desc" : "";

    // 2. Lấy tất cả tên thuộc tính của lớp Link (LinkID, LinkName, LinkURL,...)

    var properties = typeof(Link).GetProperties();

    string s = String.Empty;

    foreach (var item in properties)

    {

        // 2.1 Kiểm tra xem thuộc tính nào là virtual (public virtual Category Category...)

        var isVirtual = item.GetAccessors()[0].IsVirtual;

        // 2.2. Thuộc tính bình thường thì cho phép sắp xếp

        if (!isVirtual)

        {

            ViewBag.Headings += "<th><a href='?sortProperty=" + item.Name + "&sortOrder=" +

                ViewBag.SortOrder + "'>" + item.Name + "</a></th>";

        }

        // 2.3. Thuộc tính virtual (public virtual Category Category...) thì không sắp xếp được

        // cho nên không cần tạo liên kết

        else ViewBag.Headings += "<th>" + item.Name + "</th>";

    }

    // 3. Truy vấn lấy tất cả đường dẫn

    var links = from l in db.Links

          select l;

    // 4. Tạo thuộc tính sắp xếp mặc định là "LinkID"

    if (String.IsNullOrEmpty(sortProperty)) sortProperty = "LinkID";

    // 5. Sắp xếp tăng/giảm bằng phương thức OrderBy sử dụng trong thư viện Dynamic LINQ

    if (sortOrder == "desc") links = links.OrderBy(sortProperty + " desc");

    else links = links.OrderBy(sortProperty);

    // 6. Trả kết quả về Views

    return View(links.ToList());

}

Trong đoạn mã trên, bạn có thể thấy chú thích giải thích từng đoạn. Trước hết lưu ý nhúng 2 dòng using System.[**Linq**](https://www.dammio.com/glossary/linq).Dynamic; và using System.[**Linq**](https://www.dammio.com/glossary/linq).Expressions; để sử dụng thư viện Dynamic [LINQ](https://www.dammio.com/glossary/linq) khi truy vấn.

Sau đó, chúng ta sử dụng biến ViewBag SortOrder để giữ trạng thái sắp xếp tăng/giảm (desc là giảm và mặc định là tăng). Chúng ta tạo biến ViewBag Headings lưu trữ các tên cột (thuộc tính) của lớp Link.cs, các thuộc tính virtual đều không thể sắp xếp được (do là bảng liên kết) cho nên chúng ta loại ra, không tại đường dẫn sắp xếp.

Kế đến, bạn tạo câu truy vấn và đặt giá trị sortProperty (thuộc tính sắp xếp) mặc định là “LinkID”. Tiếp theo, thêm câu truy vấn sắp tăng hay giảm. Ở bước này, chú ý bạn phải cài thư viện [Dynamic LINQ 1.0.7](https://www.nuget.org/packages/System.Linq.Dynamic/) ở trên mới có thể sử dụng phương thức OrderBy với giá trị chuỗi truyền vào. Cuối cùng trả kết quả truy vấn về View để hiển thị.

## Sửa tập tin Views/Link/Index.cshtml

Bạn sửa tập tin này như sau. Chú ý đoạn **@Html.Raw(ViewBag.Headings)** dùng để hiển thị mã HTML được lấy từ Controller là tiêu đề các cột cần sắp kèm liên kết tương ứng.

@model IEnumerable<DammioMVC1.Models.Link>

@{

ViewBag.Title = "Index";

}

<h2>Index</h2>

<p>

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

</p>

<table class="table">

<tr>

@Html.Raw(ViewBag.Headings)

</tr>

@foreach (var item in Model) {

<tr>

<td>

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

</td>

<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.Category.CategoryID)

</td>

<td>

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

</td>

<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 })

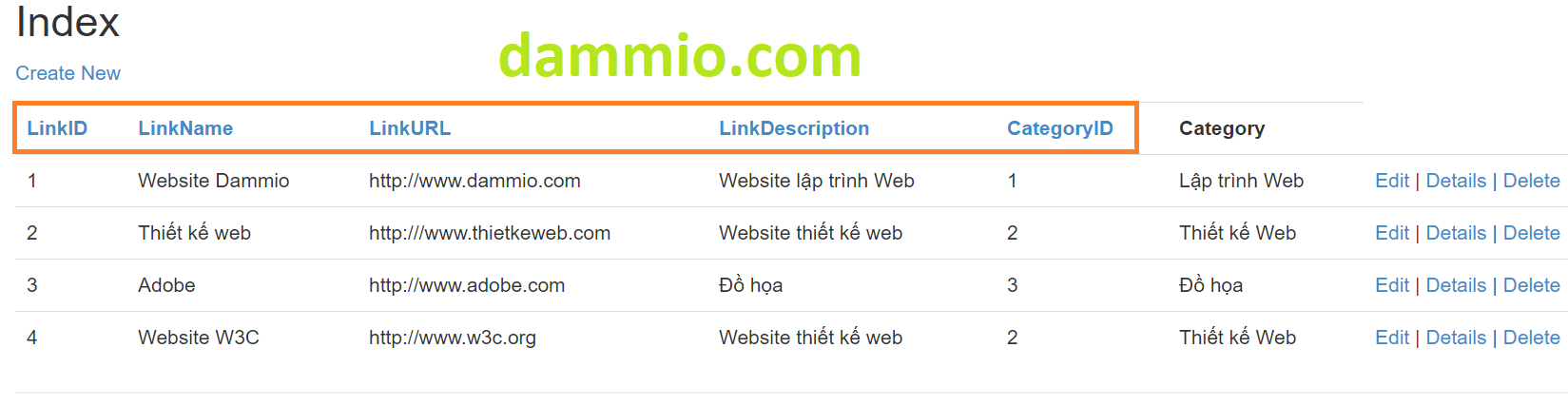
</td>

</tr>

}

</table>

Như vậy là xong, bạn mở trình duyệt chạy [URL](https://www.dammio.com/glossary/url) **http://localhost:xxxx/Link/Index** và thử sắp xếp các cột, bạn sẽ được kết quả như ý. Tuyệt vời phải không nào?



# Kết luận

Đến đây bạn đã học cách sắp xếp trong ASP.NET [MVC](https://www.dammio.com/glossary/mvc) theo nhiều cách khác nhau, tùy theo yêu cầu dự án Web bạn có thể phát triển việc sắp xếp theo nhiều cách mới lạ và độc đáo nữa. Mời bạn tiếp tục theo dõi bài tiếp theo.

# References

[1]. [DammioMVC1\_phan12.rar](https://www.dammio.com/wp-content/uploads/2018/10/DammioMVC1_phan12.rar)