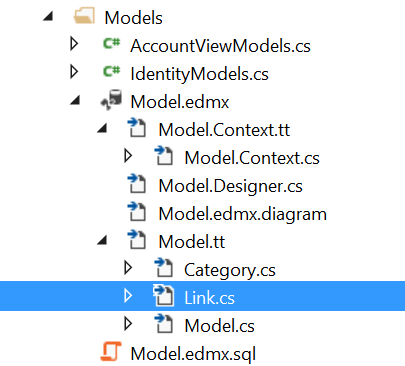
Phần 8: Tìm hiểu về các phương thức Edit và View Edit

Trong bài viết này, bạn sẽ tìm hiểu về các phương thức Edit được gieo tự động bởi Visual Studio và các view tương ứng với controller.

**Lưu ý, bài này chỉ mang tính chất tìm hiểu và có thể chứa các nội dung khá khó hiểu, vì vậy bạn cứ chậm rãi đọc và hiểu nội dung cũng như cấu trúc mã nguồn gieo bởi Visual Studio.**

Đầu tiên, bạn mở tập tin Models\Link.cs để xem nội dung. Bạn có thể xem hình sau để biết vị trí mở.



# Models\Link.cs

//------------------------------------------------------------------------------

// <auto-generated>

// This code was generated from a template.

//

// Manual changes to this file may cause unexpected behavior in your application.

// Manual changes to this file will be overwritten if the code is regenerated.

// </auto-generated>

//------------------------------------------------------------------------------

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

public string LinkName { get; set; }

public string LinkURL { get; set; }

public string LinkDescription { get; set; }

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

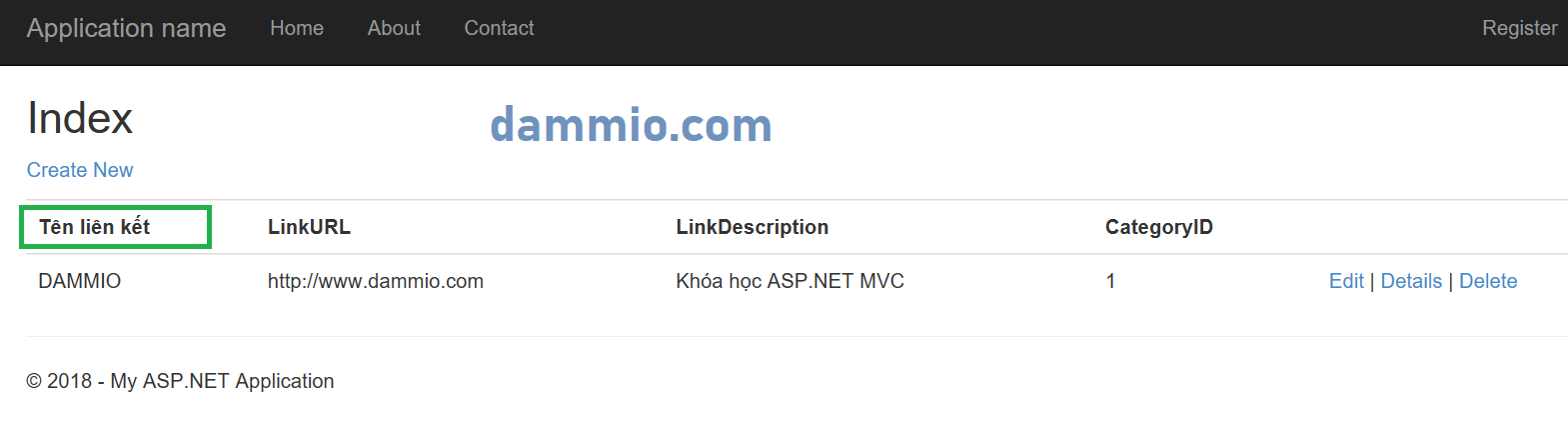
}

}

Trong đoạn mã trên, bạn để ý hai chỗ có chú thích **//thêm bằng tay**. Khi Visual Studio tự gieo ra mô hình [Entity Framework](https://www.dammio.com/glossary/entity-framework), chúng ta vẫn có thể thêm mã nguồn vào tập tin đã gieo nếu muốn. Lưu ý, phần thêm này nếu bạn gieo lại mã nguồn [EF](https://www.dammio.com/glossary/ef) lần nữa thì có thể bị mất do Visual Studio ghi đè lên. Do đó, bạn nên viết ở một tập tin riêng (phần này sẽ cập nhập hướng dẫn ở danh mục [Entity Framework](https://www.dammio.com/glossary/entity-framework) sau).

Trở lại vấn đề, đoạn mã nhúng thư viện System.ComponentModel.DataAnnotations dùng để định nghĩa chú thích cho các trường thuộc tính, bao gồm định nghĩa lại tên hiển thị, kiểu dữ liệu, định dạng dữ liệu hiển thị. Ví dụ trên định nghĩa thuộc tính **LinkName** khi xuất trên màn hình sẽ là “**Tên liên kết**“.

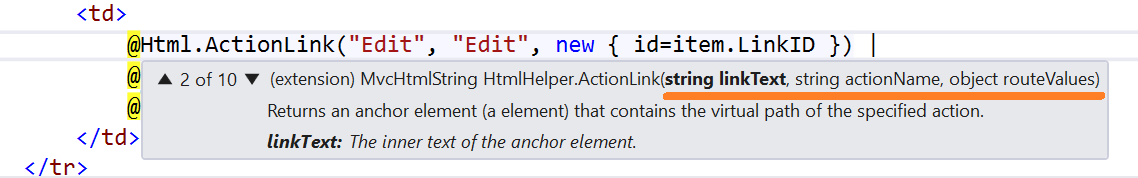
Sau đó, bạn chạy đường dẫn http://localhost:xxxx/Link/Index thì sẽ thấy tên thuộc tính này thay đổi như hình sau (ô xanh lá).



Tiếp tục, bạn mở tập tin **Views\Link\Index.cshtml** để xem mã nguồn liên kết Edit thì sẽ thấy đoạn mã.

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

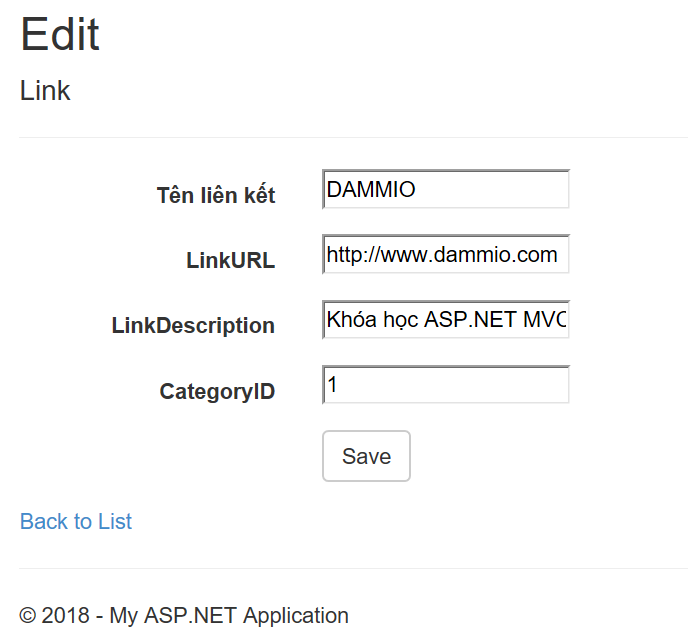
Trong đoạn mã, **@Html.ActionLink** tương ứng với thẻ <a href=””></a> chứa 3 tham số là tên đường dẫn, tên ActionName và đối tượng các giá trị như sau.



Đoạn mã trên tương ứng với đoạn mã HTML như sau, với 1 là chỉ số ID truyền vào, giá trị có thể bất kỳ.

<a href="/Link/Edit/1">Edit</a>

Sau đó, nhấp liên kết Edit trên giao diện Web để đi đến trang chỉnh sửa liên kết http://localhost:xxxx/Link/Edit/1.



Đường dẫn http://localhost:xxxx/Link/Edit/1 được cấu hình trong tập tin config tương ứng với mẫu {controller}/{action}/{id}. Bạn có thể thay đổi kiểu đường dẫn nếu muốn bằng cách chỉnh sửa tập tin này.

# Nội dung App\_Start\RouteConfig.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.Mvc;

using System.Web.Routing;

namespace DammioMVC

{

public class RouteConfig

{

public static void RegisterRoutes(RouteCollection routes)

{

routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");

routes.MapRoute(

name: "Default",

url: "{controller}/{action}/{id}",

defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }

);

routes.MapRoute(

name: "Dammio",

url: "{controller}/{action}/{ten}/{tuoi}"

);

}

}

}

Như trong đoạn mã, tôi có chỉnh thêm đường dẫn Dammio với [url](https://www.dammio.com/glossary/url): “{controller}/{action}/{ten}/{tuoi}”. Bạn có thể tùy chỉnh [URL](https://www.dammio.com/glossary/url) theo ý mình.

# Controllers/LinkController.cs

Kế đến, bạn mở lại nội dung tập tin Controllers/LinkController.cs và tìm phương thức Edit để xem nội dung. Ở đây, bạn có 2 phương thức Edit khác nhau theo kiểu quá tải hàm.

* Phương thức Edit(int? id) dùng để dò tìm chỉ số id tương ứng với bảng trong database sử dụng [Entity Framework](https://www.dammio.com/glossary/entity-framework) bằng phương thức Find, và trả về kết quả cho view Edit.
* Phương thức Edit([Bind(Include=”LinkID,LinkName,LinkURL,LinkDescription,CategoryID”)] Link link) thứ hai dùng để lấy dữ liệu đã thay đổi bởi người dùng và cập nhật thông tin lại vào database.

// GET: /Link/Edit/5

public ActionResult Edit(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Link link = db.Links.Find(id); // Tìm liên kết theo id tương ứng

if (link == null)

{

return HttpNotFound(); // Nếu không tìm thấy hiển thị kết quả không thấy

}

return View(link); // nếu tìm thấy hiển thị nội dung

}

// POST: /Link/Edit/5

// To protect from overposting attacks, please enable the specific properties you want to bind to, for

// more details see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit([Bind(Include="LinkID,LinkName,LinkURL,LinkDescription,CategoryID")] Link link)

{

if (ModelState.IsValid)

{

db.Entry(link).State = EntityState.Modified; // chọn trạng thái thay đổi

db.SaveChanges(); // lưu lại

return RedirectToAction("Index"); // sau khi cập nhật quay về trang chủ Index

}

return View(link);

}

Lưu ý phương thức Edit thứ hai dùng thuộc tính HttpPost. Điều đó có nghĩa dữ liệu cập nhật chỉ dùng POST để bảo mật. Bạn có thể áp dụng thuộc tính HttpGet vào phương thức thứ nhất, nhưng không cần thiết vì mặc định đã là như vậy. Thuộc tính Bind là cơ chế bảo mật quan trọng đề phòng hacker đẩy dữ liệu quá mức đến mô hình. Bạn chỉ có thể chứa các thuộc tính trong thuộc tính bind bạn muốn thay đổi.

# Views\Link\Edit.cshtml

Trong mô hình đơn giản bài này, chúng ta sẽ trói tất cả dữ liệu vào mô hình. @Html.AntiForgeryToken() gieo token (mã bảo mật) dùng để chống lừa đảo ở một form ẩn và token này phải khớp với phương thức Eidt trong Link controller. Thuộc tính ValidateAntiForgeryToken dùng để ngăn chặn một yêu câu lừa đảo và đi kèm với @Html.AntiForgeryToken() ở tập tin view Views\Link\Edit.cshtml với nội dung mã nguồn như sau.

@model DammioMVC.Models.Link

@{

ViewBag.Title = "Edit";

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

}

<h2>Edit</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<h4>Link</h4>

<hr />

@Html.ValidationSummary(true)

@Html.HiddenFor(model => model.LinkID)

<div class="form-group">

@Html.LabelFor(model => model.LinkName, new { @class = "control-label col-md-2" })

<div class="col-md-10">

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

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

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.LinkURL, new { @class = "control-label col-md-2" })

<div class="col-md-10">

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

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

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.LinkDescription, new { @class = "control-label col-md-2" })

<div class="col-md-10">

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

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

</div>

</div>

<div class="form-group">

@Html.LabelFor(model => model.CategoryID, new { @class = "control-label col-md-2" })

<div class="col-md-10">

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

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

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="Save" class="btn btn-default" />

</div>

</div>

</div>

}

<div>

@Html.ActionLink("Back to List", "Index")

</div>

@section Scripts {

@Scripts.Render("~/bundles/jqueryval")

}

Dựa theo đoạn mã trên, lưu ý mệnh đề @model DammioMVC.Models.Link ở đầu tập tin để mô tả mô hình mong đợi cho bản mẫu view là dạng mô hình Link.Html.LabelFor dùng để hiển thị tên của các trường thuộc tính Link như LinkName, LinkURL và LinkDescription. Html.EditorFor dùng để gieo một phần tử input cho phép người dùng nhập liệu. Html.ValidationMessageFor hiển thị thông điệp kiểm chứng liên quan đến thuộc tính, chẳng hạn ở trường nhập liệu kiểu int mà bạn nhập một chuỗi “abc” thì hệ thống sẽ báo lỗi nhập sai dữ liệu.

Tiếp theo, bạn chạy lại liên kết http://localhost:xxxx/Link/Edit/1 ở trình duyệt. Chuột phải ở trình duyệt, bạn có thể thấy mã nguồn HTML như sau.

# Mã nguồn của đường dẫn <http://localhost:xxxx/Link/Edit/1>

<form action="/Link/Edit/1" method="post"><input name="\_\_RequestVerificationToken" type="hidden" value="NF2GxIbGPoEIPVAFPTr4JNPtk9NGu7pjUR9Z5hvl5rkzHvHWn6SBjSBCk9VmnKLPdGcTIrjF2dsoZvUPG9A5wbW8\_ZPX5zttM8njfBBjJOk1" /> <div class="form-horizontal">

<h4>Link</h4>

<hr />

<input data-val="true" data-val-number="The field LinkID must be a number." data-val-required="The LinkID field is required." id="LinkID" name="LinkID" type="hidden" value="1" />

<div class="form-group">

<label class="control-label col-md-2" for="LinkName">T&#234;n li&#234;n kết</label>

<div class="col-md-10">

<input class="text-box single-line" id="LinkName" name="LinkName" type="text" value="DAMMIO" />

<span class="field-validation-valid" data-valmsg-for="LinkName" data-valmsg-replace="true"></span>

</div>

</div>

<div class="form-group">

<label class="control-label col-md-2" for="LinkURL">LinkURL</label>

<div class="col-md-10">

<input class="text-box single-line" id="LinkURL" name="LinkURL" type="text" value="http://www.dammio.com" />

<span class="field-validation-valid" data-valmsg-for="LinkURL" data-valmsg-replace="true"></span>

</div>

</div>

<div class="form-group">

<label class="control-label col-md-2" for="LinkDescription">LinkDescription</label>

<div class="col-md-10">

<input class="text-box single-line" id="LinkDescription" name="LinkDescription" type="text" value="Khóa học ASP.NET MVC" />

<span class="field-validation-valid" data-valmsg-for="LinkDescription" data-valmsg-replace="true"></span>

</div>

</div>

<div class="form-group">

<label class="control-label col-md-2" for="CategoryID">CategoryID</label>

<div class="col-md-10">

<input class="text-box single-line" data-val="true" data-val-number="The field CategoryID must be a number." id="CategoryID" name="CategoryID" type="number" value="1" />

<span class="field-validation-valid" data-valmsg-for="CategoryID" data-valmsg-replace="true"></span>

</div>

</div>

<div class="form-group">

<div class="col-md-offset-2 col-md-10">

<input type="submit" value="Save" class="btn btn-default" />

</div>

</div>

</div>

</form>

Để ý phần tử form, bạn thấy thuộc tính value có giá trị “NF2GxIbGPoEIPVAFPTr4JNPtk9N..” thì đây chính là giá trị token ẩn (tự động gieo mỗi lần duyệt Web) để bảo mật cho trang web. Nếu một hacker khi submit dữ liệu thông qua form Edit mà không gửi kèm đúng token thì không thể ghi đè vào database. Cơ chế này làm cho các ứng dụng ASP.NET [MVC](https://www.dammio.com/glossary/mvc) cực kỳ bảo mật, vì vậy rất thích hợp để áp dụng cho các hệ thống hoạt động ngân hàng hay thanh toán điện tử.

# Xử lý yêu cầu POST

Đoạn mã sau hiển thị phiên bản HttpPost của phương thức Edit.

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit([Bind(Include="LinkID,LinkName,LinkURL,LinkDescription,CategoryID")] Link link)

{

if (ModelState.IsValid)

{

db.Entry(link).State = EntityState.Modified;

db.SaveChanges();

return RedirectToAction("Index");

}

return View(link);

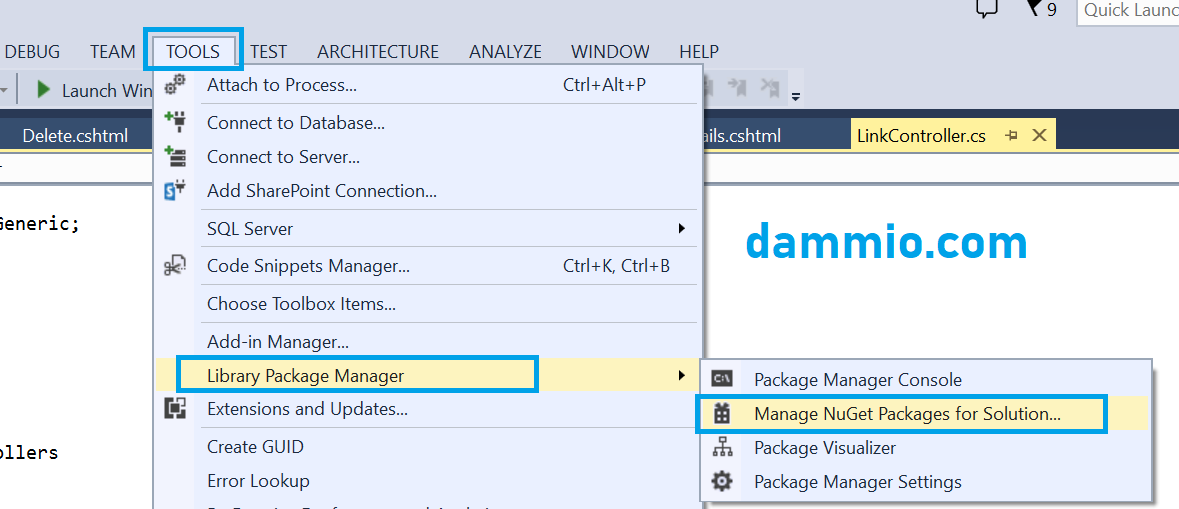
}

Như đã giải thích ở trên, thuộc tính ValidateAntiForgeryToken kiểm chứng token XSRF (Cross-site request forgery) được gieo bởi cuộc gọi @Html.AntiForgeryToken() ở view. Mô hình ASP.NET [MVC](https://www.dammio.com/glossary/mvc) lấy các giá trị cần post ở form, sau đó tạo một đối tượng Link, đẩy các giá trị này vào và kiểm chứng dữ liệu bằng phương thức **ModelState.IsValid**, cuối cùng dữ liệu được gửi đi đến server để cập nhật hay chỉnh sửa. Nếu dữ liệu đầu vào hợp lệ, dữ liệu sẽ lưu vào tập hợp Link của db (thể hiện DbContext). Dữ liệu Link được lưu vào database thông qua phương thức SaveChanges. Sau đó, đoạn mã sẽ quay về giao diện trang Index của lớp LinkController, nơi hiển thị tất cả các Link, bao gồm những thay đổi vừa xảy ra.

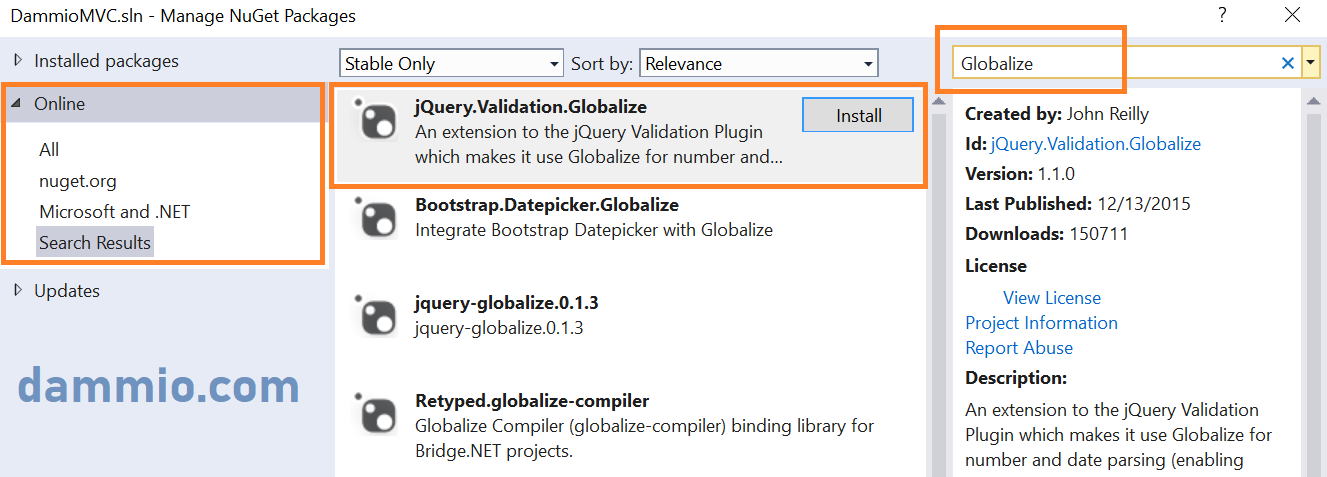
# Phiên bản Globalize (toàn cầu hóa)

Nếu bạn không có nhu cầu về vấn đề ngôn ngữ ngoài tiếng Anh, bạn có thể bỏ qua phần này. Để hỗ trợ việc kiểm chứng dữ liệu JQuery với ngôn ngữ không phải tiếng Anh, bạn phải nhúng 2 tập tin **globalize.js** và **cultures/globalize.cultures.js** (từ https://github.com/jquery/globalize) để sử dụng Globalize.parseFloat. Chẳng hạn, trong tiếng Việt khi thể hiện con số hàng nghìn, người ta dùng chấm phẩy (,) thay vì dấu chấm (.) như trong tiếng Anh. Ví dụ 1,000,000 thay vì 1.000.000 như tiếng Anh. Hay về định dạng ngày tháng, các ngôn ngữ không phải tiếng Anh đều có định dạng khác.

Bạn có thể cài đặt gói Globalize như sau. Từ menu Tools, chọn NuGetLibrary Package Manager, sau đó chọn Manage NuGet Packages for Solution.



Kế đến ở thanh bên trái, chọn Online, ở hộp tìm kiếm bên góc phải, gõ Globalize, sau đó chọn gói này và cài đặt.



Đợi vài phút để Visual Studio cài đặt cho bạn, sau khi cài xong, bạn copy đoạn mã sau đặt cuối trang Views/Link/Edit.cshtml nếu Visual Studio không tự động thêm cho bạn.

@section Scripts {

@Scripts.Render("~/bundles/jqueryval")

<script src="~/Scripts/globalize/globalize.js"></script>

<script src="~/Scripts/globalize/cultures/globalize.culture.@(System.Threading.Thread.CurrentThread.CurrentCulture.Name).js"></script>

<script>

$.validator.methods.number = function (value, element) {

return this.optional(element) ||

!isNaN(Globalize.parseFloat(value));

}

$(document).ready(function () {

Globalize.culture('@(System.Threading.Thread.CurrentThread.CurrentCulture.Name)');

});

</script>

<script>

jQuery.extend(jQuery.validator.methods, {

range: function (value, element, param) {

//Use the Globalization plugin to parse the value

var val = Globalize.parseFloat(value);

return this.optional(element) || (

val >= param[0] && val <= param[1]);

}

});

$.validator.methods.date = function (value, element) {

return this.optional(element) ||

Globalize.parseDate(value) ||

Globalize.parseDate(value, "yyyy-MM-dd");

}

</script>

}

Để tránh thêm đoạn mã vào mỗi Edit view, bạn có thể di chuyển đoạn mã đến tập tin layout, chẳng hạn như **\_Layout.cshtml**. Nếu bạn muốn ép máy tính chỉ dùng tiếng Anh US, bạn có thể thêm phần tử toàn cầu hóa vào tập tin **Web.config**, theo đoạn mã sau.

<system.web>

<globalization culture ="en-US" />

<!--elements removed for clarity-->

</system.web>

1. **Kết luận**

Bài viết này đã hướng dẫn và giải thích chi tiết nhất có thể về các phương thức Edit, cũng như các View liên quan. Về các phương thức khác như Create, Delete, mời bạn theo dõi thêm các phần tiếp theo.

[1]. [DammioMVC\_Phan8.rar](https://www.dammio.com/wp-content/uploads/2018/07/DammioMVC_Phan8.rar)