# Bài thực hành 3: XÂY DỰNG TRANG ADMIN CHO CATEGORIES

## tạo controller có sử dụng entity framework

R\_Click Areas\Admin\Controller\Add Controller

Installed\Common\MVC\Controller\MVC 5 Controller with view using Entity Framework\Add

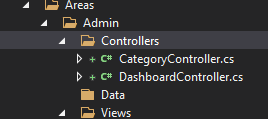
Model Class = Categories (MyClassModel)

Data Context Class = MyDBContext(MyClass.Model)

🗹 Generate View

🗹 Use a layout page: ~/Areas/Admin/Views/Shared/\_LayoutAdmin.cshtml

Controller name = CategoryController[[1]](#footnote-1)



Nội dung của controller:

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.Entity;

using System.Linq;

using System.Net;

using System.Web;

using System.Web.Mvc;

using MyClass.Model;

namespace UDW.Areas.Admin.Controllers

{

public class CategoryController : Controller

{

private MyDBContext db = new MyDBContext();

// GET: Admin/Category

public ActionResult Index()

{

...

}

// GET: Admin/Category/Details/5

public ActionResult Details(int? id)

{

...

return View(categories);

}

// GET: Admin/Category/Create

public ActionResult Create()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create([Bind(Include = "Id,Name,Slug,ParentID,Order,MetaDesc,MetaKey,CreateAt,CreateBy,UpdateAt,UpdateBy,Status")] Categories categories)

{

...

return View(categories);

}

// GET: Admin/Category/Edit/5

public ActionResult Edit(int? id)

{

...

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit([Bind(Include = "Id,Name,Slug,ParentID,Order,MetaDesc,MetaKey,CreateAt,CreateBy,UpdateAt,UpdateBy,Status")] Categories categories)

{

...

return View(categories);

}

// GET: Admin/Category/Delete/5

public ActionResult Delete(int? id)

{

...

return View(categories);

}

// POST: Admin/Category/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

...

return RedirectToAction("Index");

}

protected override void Dispose(bool disposing)

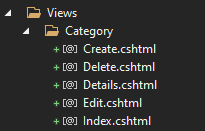
{

...

}

}

Có thể nhận ra những đoạn mã sinh ra được sử dụng để thực hiện các thao tác cơ bản truy vấn cơ sở dữ liệu (CRUD). Tương ứng mỗi phương thức (Index, Edit…) sẽ có 1 View tương ứng (Goto View)



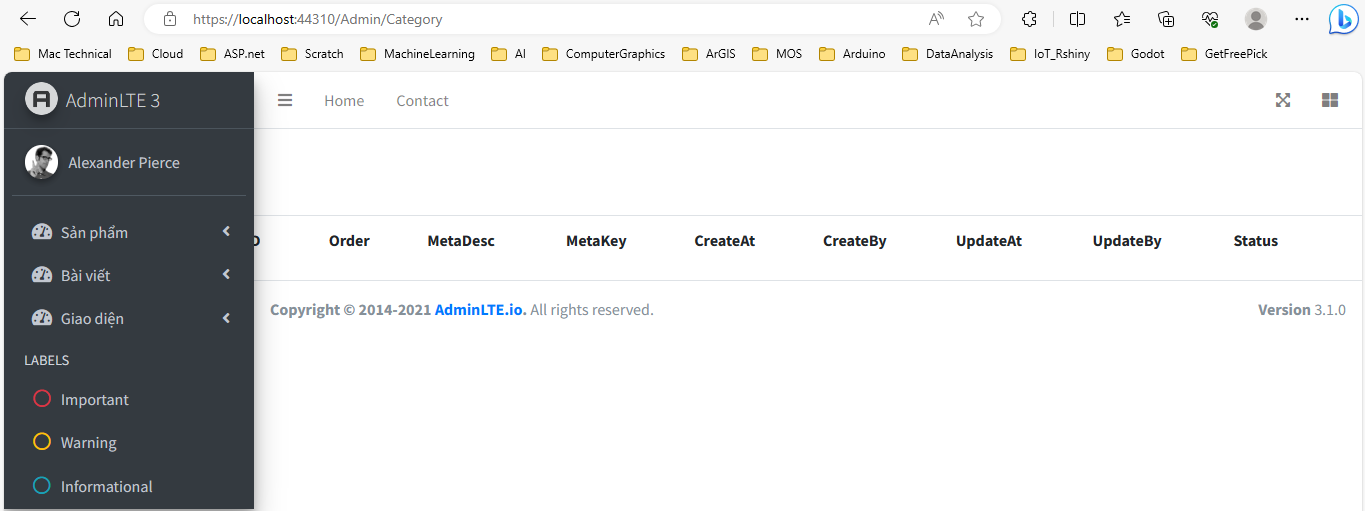
### Build - RUN

Liên kết thực thi đến trang Index của Category: <https://localhost:44310/admin/category>

Admin: Folder: Admin

Category: Controller

Index: Không cần ghi vì đó là giá trị mặc định của



## Chỉnh sửa trang hiển thị

Sử dụng trang INDEX của Dashboard thay thế cho đoạn <h2>Index</h2>

Trở thành

<!-- Content Wrapper. Contains page content -->

<div class="content-wrapper">

<!-- Main content -->

<section class="content my-2">

<!-- Default box -->

<div class="card">

<div class="card-header">

<div class="row">

<div class="col-md-6">

<strong>TITLE</strong>

</div>

<div class="col-md-6 text-right">

LINK

</div>

</div>

</div>

<div class="card-body">

NOI DUNG

<!-- /.card-body -->

</div>

</div>

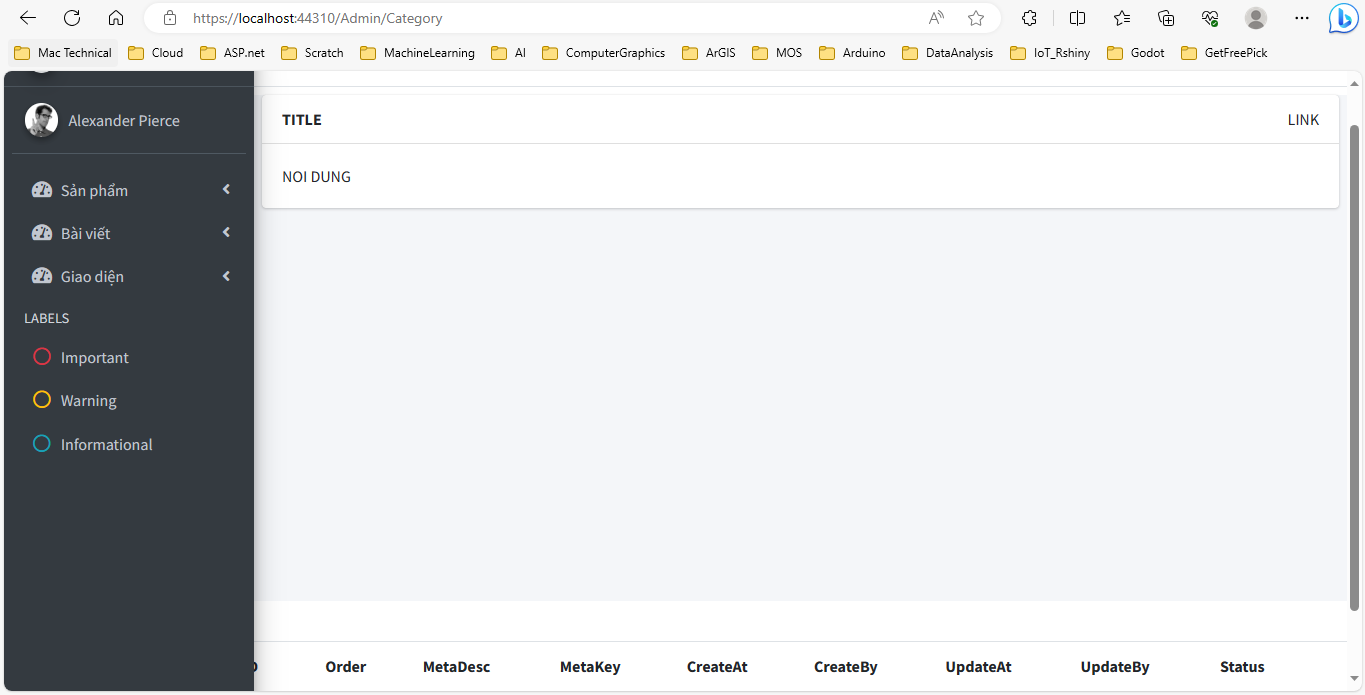
<!-- /.card -->

</section>

<!-- /.content -->

</div>

<!-- /.content-wrapper -->

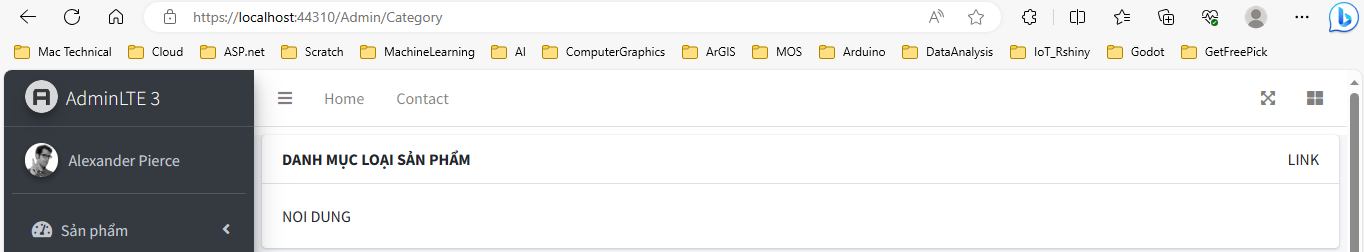


### Edit Title

<div class="col-md-6">

<strong>DANH MỤC LOẠI SẢN PHẨM</strong>

</div>



### Chỉnh sửa phần nội dung

Thay thế từ NOI DUNG đoạn sau:

<div class="card-body">

NOI DUNG

<!-- /.card-body -->

</div>

Thành nội dung của đoạn <table class="table">...</table>

<div class="card-body">

<table class="table">

<tr>

<th>

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

</th>

...

<th>

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

</th>

<th></th>

</tr>

@foreach (var item in Model)

{

<tr>

<td>

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

</td>

...

<td>

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

</td>

<td>

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

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

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

</td>

</tr>

}

</table>

<!-- /.card-body -->

</div>

### Thay thế LINK

Thay thế nội dung LINK trong đoạn:

<div class="col-md-6 text-right">

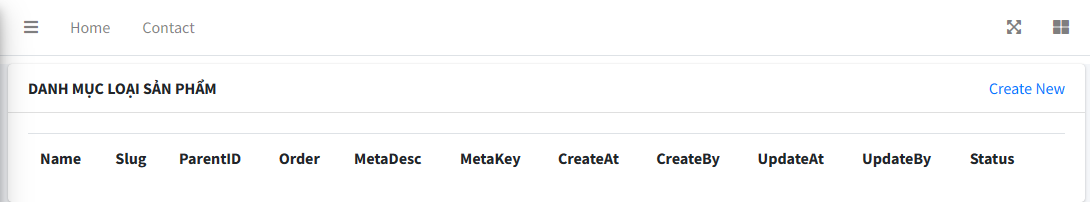
LINK

</div>

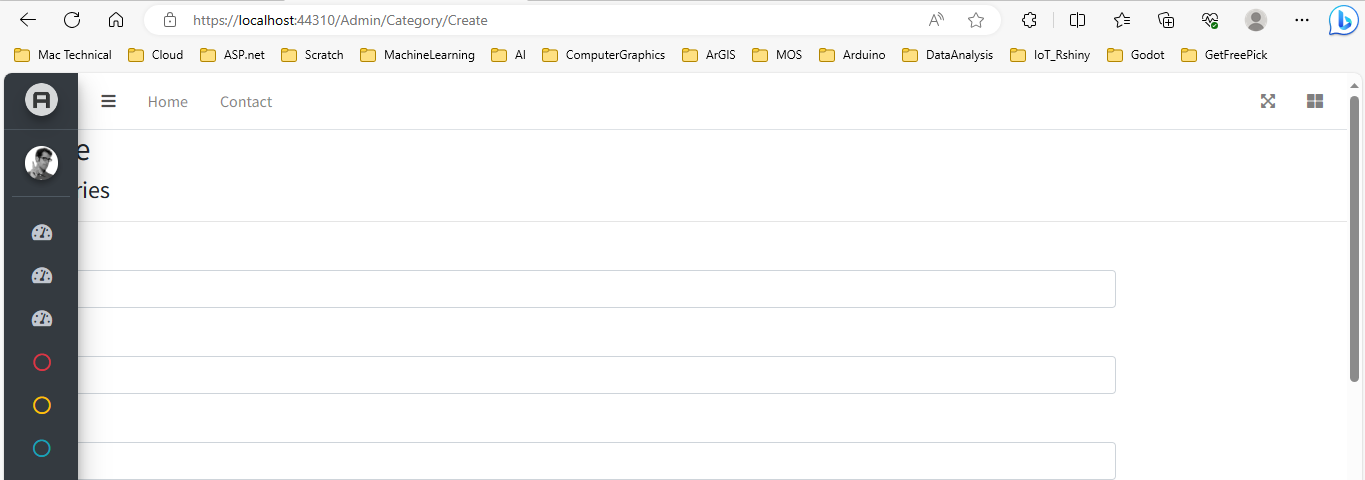
Thành

<div class="col-md-6 text-right">

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

</div>

# Tạo mới loại sản phẩm



Nội dung trang Create sau khi Click nút Create ở trang Index hoặc <https://localhost:44310/Admin/Category/Create>

Đi đến đoạn code sau trong Category Controller

// GET: Admin/Category/Create

public ActionResult Create()

{

return View();

}

R\_Click Create - Goto View

## Sao chép nội dung trang INDEX của Admin\Dashboard

<!-- Content Wrapper. Contains page content -->

<div class="content-wrapper">

...

</div>

<!-- /.content-wrapper -->

Và thay thế cho đoạn của Create.cshtml: <h2>Create</h2>

Thay thế nội dung của LINK

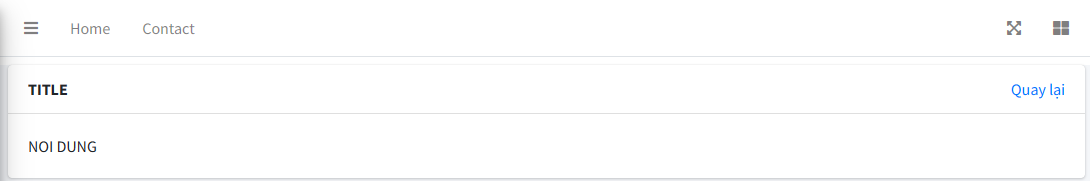
<div class="col-md-6 text-right>LINK</div>

Thành

<div class="col-md-6 text-right">

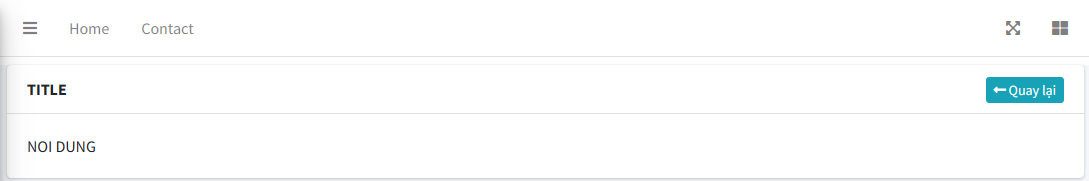
@Html.ActionLink("Quay lại", "Index")

</div>



Ta có thể thay thế bằng đoạn mã sau:

<a href="@Url.Action("Index", "Category")" class="btn-sm btn-info"> <i class="fa fa-long-arrow-alt-left"></i> Quay lại</a>



#### Như vậy để có thể đi đến một liên kết ta có 2 phương thức:

@Html.ActionLink("Tên hiển thị", "phương thức")

<a href="@Url.Action("ACTION", "CONTROLLER")" </a>

## Chỉnh sửa trang

### Form thiết kế

Toàn bộ nội dung của trang CREATE là một form, do vậy ta cần khai báo:

@model MyClass.Model.Categories

@{

ViewBag.Title = "Create";

Layout = "~/Areas/Admin/Views/Shared/\_LayoutAdmin.cshtml";

}

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<!-- Content Wrapper. Contains page content -->

<div class="content-wrapper">

...

</div>

<!-- /.content-wrapper -->

}

### Trang báo lỗi

@Html.AntiForgeryToken()

@Html.ValidationSummary(true, "", new { @class = "text-danger" })

### Chỉnh sửa phần NOI DUNG

<div class="card-body">

NOI DUNG

<!-- /.card-body -->

</div>

Thay thế NOI DUNG thành các đoạn có cú pháp:

<div class="form-group">

@Html.LabelFor()

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

@Html.EditorFor()

@Html.ValidationMessageFor()

</div>

</div>



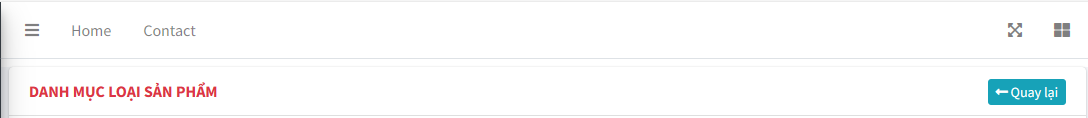
### Chỉnh sửa Title

<div class="col-md-6">

<strong>TITLE</strong>

</div>

Sửa TITLE thành <strong class="text-danger">DANH MỤC LOẠI SẢN PHẨM</strong>



### Chỉnh sửa nút CREATE

<div class="form-group">

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

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

</div>

</div>



**Thành**

<button class="btn-sm btn-success" type="submit" name="create"> Lưu [Thêm] <i class="fa fa-save"></i> </button>



#### Thử bấm nút Thêm, ta sẽ có các báo lỗi tương ứng với trường dữ liệu bị bỏ trống



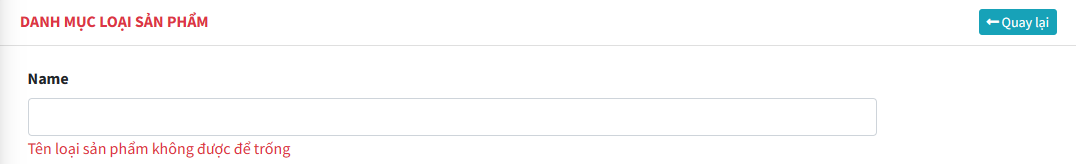
## Thay thế đoạn báo lỗi thành Tiếng Việt

### Mở nội dung Category.cs thêm nội dụng cho mục có trường bắt buộc

[Required(ErrorMessage ="Tên loại sản phẩm không được để trống")]

[Required(ErrorMessage = "Phần mô tả không được để trống")]

[Required(ErrorMessage = "Phần từ khóa không được để trống")]



Tại sao trường: CreateAt, CreateBy, ParentID, UpdateAt/By, Status không thực hiện? Vì các trường này sinh tự động (thời gian, tên người dùng) hoặc tự động chọn trong danh sách

#### Chi tiết xem ở đây

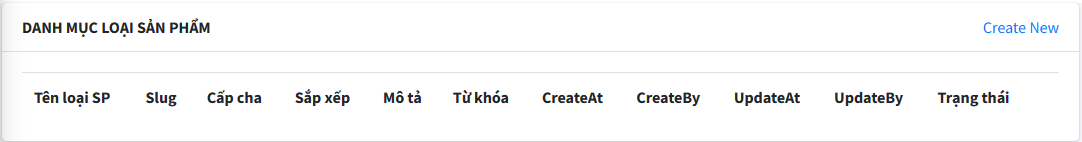
[ValidationMessageFor Method - ASP.NET MVC (tutorialsteacher.com)](https://www.tutorialsteacher.com/mvc/htmlhelper-validationmessagefor)

## Việt hóa cho các trường dữ liệu

Sử dụng câu lệnh sau: [Display(Name = "Alias")] trước khi khai báo trường dữ liệu

[Display(Name = "Từ khóa")]

public string MetaKey { get; set; }



## làm gọn trang index

### Tiêu đề của trang hiển thị

Có thể nhận thấy trang Index chứa quá nhiều thông tin không cần thiết: CreateAt, CreateBy, UpdateAt/By… chúng ta có thể ẩn bớt các trường này như sau:

<tr>

<th>@Html.DisplayNameFor(model => model.Name)</th>

<th>@Html.DisplayNameFor(model => model.Slug)</th>

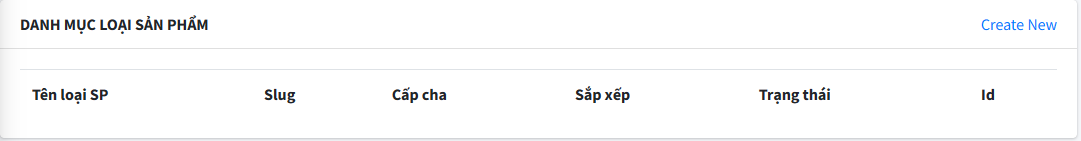
<th>@Html.DisplayNameFor(model => model.ParentID)</th>

<th>@Html.DisplayNameFor(model => model.Order)</th>

<th>@Html.DisplayNameFor(model => model.Status)</th>

<th>@Html.DisplayNameFor(model => model.Id)</th>

</tr>

Xóa các trường không cần thiết và chèn thêm trường ID vào trang hiển thị  


### Nội dung các trường dữ liệu

Chúng ta sử dụng mẫu của MVC5 >@Html.DisplayFor(...)

<td>@Html.DisplayFor(modelItem => item.Status)</td>

Hoặc sử dụng biến item ngay trong @foreach(var item in Model)

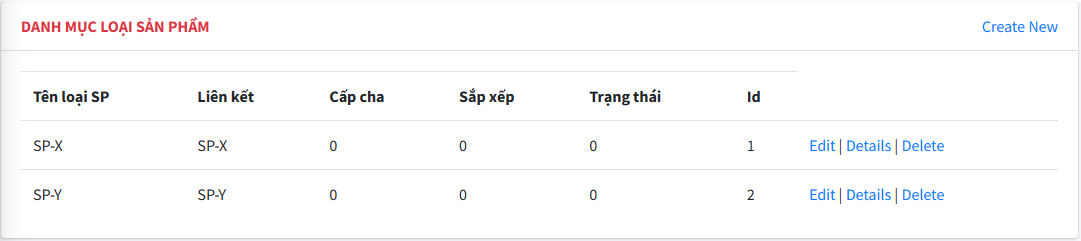
<td>@item.Id</td>

## Nhập dữ liệu cho trang CREATE

Ví dụ nhập dữ liệu cho các trường của trang Create như sau:

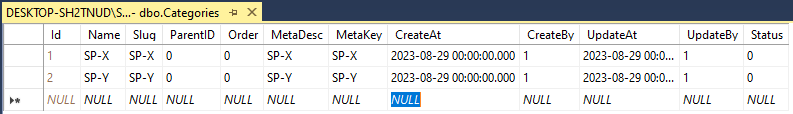
|  |  |
| --- | --- |
|  |  |

### Kết quả:



#### Tuy nhiên không phải trường dữ liệu nào cũng cần điền bằng tay: Slug, CreateBy, CreateAt, UpdateBy, UpdateAt ta sẽ bàn sau của phần này

### Nhìn lại SQL



## Chỉnh sửa trang INDEX

### Định dạng tiêu đề bảng dữ liệu

<table class="table table-bordered table-striped">

<thead>

<tr>

<th>@Html.DisplayNameFor(model => model.Name)</th>

<th>@Html.DisplayNameFor(model => model.Slug)</th>

<th>@Html.DisplayNameFor(model => model.ParentID)</th>

<th>@Html.DisplayNameFor(model => model.Order)</th>

<th>@Html.DisplayNameFor(model => model.Status)</th>

<th>@Html.DisplayNameFor(model => model.Id)</th>

</tr>

</thead>

### Phần bên dưới (các dòng dữ liệu)

<tbody>

@foreach (var item in Model)

{

<tr>

<td>@Html.DisplayFor(modelItem => item.Name)</td>

<td>@Html.DisplayFor(modelItem => item.Slug)</td>

<td>@Html.DisplayFor(modelItem => item.ParentID)</td>

<td>@Html.DisplayFor(modelItem => item.Order)</td>

<td>@Html.DisplayFor(modelItem => item.Status)</td>

<td>@item.Id</td>

<td>

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

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

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

</td>

</tr>

}

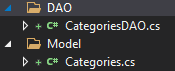
</tbody>

# Chỉnh sửa một số hàm truy vấn cơ bản (CRUD) cho categories

## Tạo lớp chuyên dùng để xử lý dữ liệu (DaO)

R\_Click DAO - Add New Item - Class

Name = CategoriesDAO[[2]](#footnote-2)



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace MyClass.DAO

{

class CategoriesDAO

{

}

}

# TRUY VẤN TOÀN BỘ DỮ LIỆU TRONG BẢNG CATEGORIES

## Chỉnh sửa file CategoriesDAO

### Khai báo thư viện

Trong tập tin CategoriesDAO nếu muốn truy vấn đến CSDL, ta cần khai báo thư viện là tập tin MyDBContext.cs. Do vậy,

using MyClass.Model[[3]](#footnote-3);

Sao chép đoạn code: private MyDBContext db = new MyDBContext();

Bên tập tin CategoriesController và bỏ bên này

namespace MyClass.DAO

{

public class CategoriesDAO

{

private MyDBContext db = new MyDBContext();

}

}

### Xây dựng hàm truy vấn toàn bộ bảng Categories

// Tra ve danh sach cac mau tin

private MyDBContext db = new MyDBContext();

//Hien thi danh sach toan bo Loai san pham: SELCT \* FROM Categories

public List<Categories> getList[[4]](#footnote-4)()

{

return db.Categories.ToList()[[5]](#footnote-5);

}

}

Mở rộng hàm truy vấn trên với điều kiện đầu vào như sau:

namespace MyClass.DAO

{

public class CategoriesDAO

{

private MyDBContext db = new MyDBContext();

/////////////////////////////////////////////////////////////////////////////////////

//Hien thi danh sach toan bo Loai san pham: SELCT \* FROM

public List<Categories> getList(string status = "All")

{

List<Categories> list = null;

switch (status)

{

case "Index":

{

list = db.Categories

.Where(m => m.Status != 0)

.ToList();

break;

}

case "Trash":

{

list = db.Categories

.Where(m => m.Status == 0)

.ToList();

break;

}

default:

{

list = db.Categories.ToList();

break;

}

}

return list;

}

}

## Chỉnh sửa file CategoriesController

### Thêm thư viện của lớp DAO vừa thiết kế

using MyClass.DAO;

### Xóa bỏ dòng:

private MyDBContext db = new MyDBContext();



Kết quả sẽ có 11 lỗi tạo ra, lý do mất hàm db kế thừa từ MyDBContext

### Tạo một biến mới: categoryDAO

CategoriesDAO categoryDAO = new CategoriesDAO();

### Chỉnh sửa hàm hiển thị toàn bộ loại sản phẩm

/////////////////////////////////////////////////////////////////////////////////////

// Admin/Category/Index: Tra ve danh sach cac mau tin

public ActionResult Index()

{

return View(categoryDAO.getList("Index"));//hien thi toan bo danh sach loai SP

}

# HIỂN THỊ NỘI DUNG CHI TIẾT CỦA TỪNG MỤC LOẠI SẢN PHẨM (1 mẫu tin)

## Chỉnh sửa file CategoriesDAO

### Xây dựng hàm trả về danh sách 1 mẫu tin

/////////////////////////////////////////////////////////////////////////////////////

//Hien thi danh sach 1 mau tin (ban ghi)

public Categories getRow(int? id[[6]](#footnote-6))

{

if (id == null)

{

return null;

}

else

{

return db.Categories.Find(id);

}

}

## Chỉnh sửa file CategoriesController

Thay thế hàm: Categories categories = db.Categories.Find(id);

Thành categoryDAO.getRow(id);

/////////////////////////////////////////////////////////////////////////////////////

// Admin/Category/Detail: Hien thi mot mau tin

public ActionResult Details(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Categories categories = categoryDAO.getRow(id);

if (categories == null)

{

return HttpNotFound();

}

return View(categories);

}

Tương tự thay thế 4 chỗ kế tiếp: 

# Tạo mới một mẫu tin

## Chỉnh sửa file CategoriesDAO

### Xây dựng hàm thêm mới 1 mẫu tin

/////////////////////////////////////////////////////////////////////////////////////

///Them moi mot mau tin

public int Insert(Categories row)

{

db.Categories.Add(row);

return db.SaveChanges()[[7]](#footnote-7);

}

## Chỉnh sửa file CategoriesController

/////////////////////////////////////////////////////////////////////////////////////

// GET: Admin/Category/Create: Them moi mot mau tin

if (ModelState.IsValid)

{

categoryDAO.Insert(categories);

~~db.Categories.Add(categories);~~

~~db.SaveChanges();~~

return RedirectToAction("Index");

}

### Làm gọn tên hàm:

public ActionResult Create([Bind(Include = "Id,Name,Slug,ParentID,Order,MetaDesc,MetaKey,CreateAt,CreateBy,UpdateAt,UpdateBy,Status")] Categories categories)

#### Trở thành

/////////////////////////////////////////////////////////////////////////////////////

// GET: Admin/Category/Create: Them moi mot mau tin

public ActionResult Create()

{

return View();

}

// POST: Admin/Category/Create

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create(Categories categories)

{

if (ModelState.IsValid)

{

categoryDAO.Insert(categories);

return RedirectToAction("Index");

}

return View(categories);

}



# Cập nhật mẫu tin

## Chỉnh sửa file CategoriesDAO

### Xây dựng hàm cập nhật 1 mẫu tin

/////////////////////////////////////////////////////////////////////////////////////

///Cap nhat mot mau tin

public int Update(Categories row)

{

db.Entry(row).State = EntityState[[8]](#footnote-8).Modified;

return db.SaveChanges();

}

Các câu lệnh được sao chép từ bên CategoriesController và chỉnh sửa cho phù hợp

## Chỉnh sửa file CategoriesController

/////////////////////////////////////////////////////////////////////////////////////

// GET: Admin/Category/Edit/5: Cap nhat mau tin

public ActionResult Edit(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Categories categories = categoryDAO.getRow(id);

if (categories == null)

{

return HttpNotFound();

}

return View(categories);

}

// POST: Admin/Category/Edit/5

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Edit(Categories categories)

{

if (ModelState.IsValid)

{

categoryDAO.Update(categories);

return RedirectToAction("Index");

}

return View(categories);

}



# XÓA mẫu tin

## Chỉnh sửa file CategoriesDAO

### Xây dựng hàm xóa 1 mẫu tin ra khỏi CSDL (xóa hẳn)

/////////////////////////////////////////////////////////////////////////////////////

///Xoa mot mau tin Xoa ra khoi CSDL

public int Delete(Categories row)

{

db.Categories.Remove(row);[[9]](#footnote-9)

return db.SaveChanges();

}

## Chỉnh sửa file CategoriesController

/////////////////////////////////////////////////////////////////////////////////////

// GET: Admin/Category/Delete/5:Xoa mot mau tin ra khoi CSDL

public ActionResult Delete(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Categories categories = categoryDAO.getRow(id);

if (categories == null)

{

return HttpNotFound();

}

return View(categories);

}

// POST: Admin/Category/Delete/5:Xoa mot mau tin ra khoi CSDL

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public ActionResult DeleteConfirmed(int id)

{

Categories categories = categoryDAO.getRow(id);

categoryDAO.Delete(categories);

return RedirectToAction("Index");

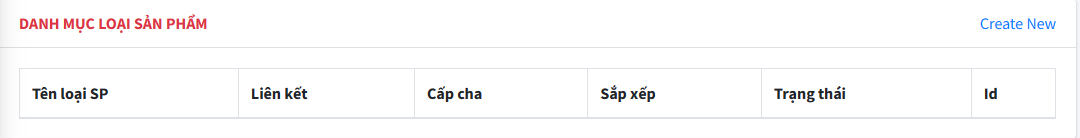
}



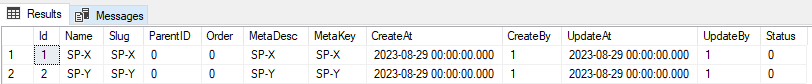
Xóa hàm: protected override void Dispose(bool disposing)



# THỬ NGHIỆM



#### ứng với:



### Xét CategoriesDAO:

public List<Categories> getList(string status = "All")

{

List<Categories> list = null;

switch (status)

{

case "Index":

{

list = db.Categories

.Where(m => m.Status != 0)

.ToList();

break;

}

case "Trash":

{

list = db.Categories

.Where(m => m.Status == 0)

.ToList();

break;

}

default:

{

list = db.Categories.ToList();

break;

}

}

return list;

}

### Xét CategoriesController:

// Admin/Category/Index: Tra ve danh sach cac mau tin

public ActionResult Index()

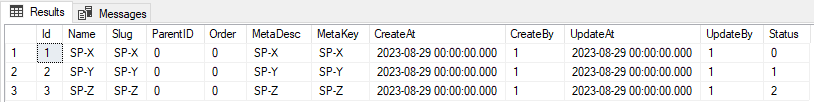
{

return View(categoryDAO.getList("Index"));//hien thi toan bo danh sach loai SP

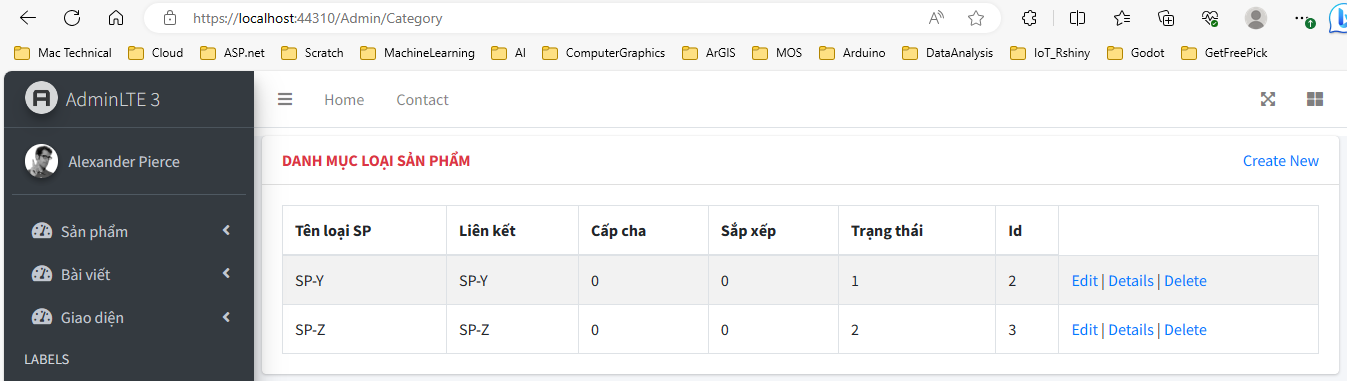
}

#### Rõ ràng Index ứng với staus ==1

## Thêm và chỉnh sửa DL trong SQL



## Kiểm tra lại kết quả trang Index



1. Đây là đường link của controller tránh nhầm lẫn với tên của Tables [↑](#footnote-ref-1)
2. Tên lớp = Tên bảng trong Model [↑](#footnote-ref-2)
3. Folder Model bên trong Myclass Project [↑](#footnote-ref-3)
4. Hàm có tên: getList [↑](#footnote-ref-4)
5. Copy từ CategoriesController [↑](#footnote-ref-5)
6. Tại sao hàm có giá trị int? id, hãy xem hàm Detail trong Categories.cs [↑](#footnote-ref-6)
7. Được sử dụng từ file CategoriesController.cs [↑](#footnote-ref-7)
8. Cần sử dụng thư viện: using System.Data.Entity; [↑](#footnote-ref-8)
9. Sao chép từ hàm Delete của CategoriesController.cs [↑](#footnote-ref-9)