MVC – JUAN CARLOS ZULUAGA

Contents

[CRUD 2](#_Toc460580305)

[MANUAL MIGRATION 5](#_Toc460580306)

[AUTOMATIC MIGRATIONS 6](#_Toc460580307)

[ONE TO MANY RELATIONSHIP 6](#_Toc460580308)

[DATA ANOTATIONS 7](#_Toc460580309)

[MANY TO MANY RELATIONSHIP 8](#_Toc460580310)

[DISABLE CASCADE DELETING RULE 9](#_Toc460580311)

[MENUS AND SUBMENUS 9](#_Toc460580312)

[AJAX CALLS 10](#_Toc460580313)

[WEB API & AJAX CALLS 11](#_Toc460580314)

[SECURITY 14](#_Toc460580315)

[LOGIN 14](#_Toc460580316)

[PERMISIONS 14](#_Toc460580317)

[CREATE ROLES AND SUPER USER 14](#_Toc460580318)

[MASTER DETAIL 17](#_Toc460580319)

[MODELS 17](#_Toc460580320)

[CONTROLLERS 19](#_Toc460580321)

[VIEWS 23](#_Toc460580322)

[PUBLISH THE APP 25](#_Toc460580323)

[UPLOAD AND SHOW IMAGES 25](#_Toc460580324)

[CREATE ROLE, USER AND ASSING ROLE TO USER 26](#_Toc460580325)

[MASTER DETAIL FORM 27](#_Toc460580326)

[ENABLE WEB TRACE 32](#_Toc460580327)

[CRYSTAL REPORTS IN APP PUBLISHED 32](#_Toc460580328)

[CASCADE DROPDOWNS 32](#_Toc460580329)

[RECOVER & EMAIL PASSWORD 34](#_Toc460580330)

[PAGING 35](#_Toc460580331)

[FILTERS IN VIEW 37](#_Toc460580332)

[MAIL HELPER 40](#_Toc460580333)

[USER HELPER 41](#_Toc460580334)

[USER MANAGEMENT 43](#_Toc460580335)

[TO IDENTIFICATE MODEL ERRORS 45](#_Toc460580336)

[IMPROVE FILE PICKER 45](#_Toc460580337)

[GENERIC CASCADE DROP DOWN LIST 47](#_Toc460580338)

[GENERIC VALIDATIONS FOR DUPLICATES AND DELETES 49](#_Toc460580339)

[DATE PICKERS 50](#_Toc460580340)

[MODAL DIALOGS 51](#_Toc460580341)

[JOIN SAMPLES 53](#_Toc460580342)

[LOGO IN MENU 54](#_Toc460580343)

[DELETE CONFIRM WITHOUT ALERT 55](#_Toc460580344)

# CRUD

1. Create the model:

public class Product

{

[Key]

public int ProductID { get; set; }

public string Description { get; set; }

public decimal Price { get; set; }

public DateTime LastBuy { get; set; }

public float Stock { get; set; }

}

1. Create folder Context and class Store Context:

public class StoreContext: DbContext

{

public DbSet<Product> Products { get; set; }

}

1. Rebuild the project
2. Change the connection string:

<add name="StoreContext"

connectionString="Data Source=.;Initial Catalog=Z-Market;Integrated Security=True"

providerName="System.Data.SqlClient" />

1. Add the Product Controller with read write actions
2. In the Index method:

private StoreContext db = new StoreContext();

// GET: Product

public ActionResult Index()

{

return View(db.Products.ToList());

}

1. In the Create method:

// GET: Product/Create

[HttpGet]

public ActionResult Create()

{

return View();

}

// POST: Product/Create

[HttpPost]

public ActionResult Create(Product product)

{

try

{

if (ModelState.IsValid)

{

db.Products.Add(product);

db.SaveChanges();

return RedirectToAction("Index");

}

return View(product);

}

catch

{

return View(product);

}

}

1. Change the route config:

routes.MapRoute(

name: "Default",

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

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

);

1. In the Details method:

// GET: Products/Details/5

public ActionResult Details(int? id)

{

if(id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

var product = db.Products.Find(id);

if(product == null)

{

return HttpNotFound();

}

return View(product);

}

1. In Edit method:

// GET: Products/Edit/5

public ActionResult Edit(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

var product = db.Products.Find(id);

if (product == null)

{

return HttpNotFound();

}

return View(product);

}

// POST: Products/Edit/5

[HttpPost]

public ActionResult Edit(Product product)

{

try

{

if (ModelState.IsValid)

{

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

db.SaveChanges();

return RedirectToAction("Index");

}

return View(product);

}

catch

{

return View(product);

}

}

1. In Delete method:

// GET: Products/Delete/5

public ActionResult Delete(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

var product = db.Products.Find(id);

if (product == null)

{

return HttpNotFound();

}

return View(product);

}

// POST: Products/Delete/5

[HttpPost]

public ActionResult Delete(int id, Product product)

{

try

{

if (ModelState.IsValid)

{

product = db.Products.Find(id);

if (product == null)

{

return HttpNotFound();

}

db.Products.Remove(product);

db.SaveChanges();

return RedirectToAction("Index");

}

return View(product);

}

catch

{

return View(product);

}

}

# MANUAL MIGRATION

1. Try to add a new field to model and run it the project:

public string Remarks { get; set; }

Do you get an error, right?

1. Reverse the change and enter the following command by Package Manager Console:

Enable-Migrations -ContextTypeName StoreContext

1. Make the model change again

public string Remarks { get; set; }

1. Run the command:

Add-Migration AddRemarks

1. Run the command:

Update-Database

1. Run the project!
2. Modify the controller and views to see the new field

# AUTOMATIC MIGRATIONS

1. Modify the model and try to access to the controller, a new error appears.
2. Run the following command:

Enable-Migrations -ContextTypeName StoreContext -EnableAutomaticMigrations –Force

1. Add the following line to the Configuration.cs in Migrations folder:

AutomaticMigrationDataLossAllowed = true;

1. In the Global.asax, add the following line:

Database.SetInitializer(new MigrateDatabaseToLatestVersion<Models.StoreContext, Migrations.Configuration>());

Ojo para revisar luego:

<https://stackoverflow.com/questions/18126711/enable-migrations-with-context-in-separate-assembly>

# ONE TO MANY RELATIONSHIP

1. Crete the new model Employee:

public class Employee

{

[Key]

public int EmployeeID { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public decimal Salary { get; set; }

public float BonusPercent { get; set; }

public DateTime DateOfBirth { get; set; }

public DateTime StartTime { get; set; }

public string EMail { get; set; }

public string URL { get; set; }

public int DocumentTypeID { get; set; }

public virtual DocumentType DocumentType { get; set; }

}

1. Create the new model Document Type:

public class DocumentType

{

[Key]

public int DocumentTypeID { get; set; }

public string Description { get; set; }

public virtual ICollection<Employee> Employees { get; set; }

}

1. Rebuild the application and create the controllers (MVC 5 Controller with views, using Entity Framework) for Document Type and Employee
2. Add those links to menu

<li>@Html.ActionLink("Products", "Index", "Products")</li>

<li>@Html.ActionLink("Document Types", "Index", "DocumentTypes")</li>

<li>@Html.ActionLink("Employees", "Index", "Employees")</li>

1. Run the project and add some records, then see the database

# DATA ANOTATIONS

1. Modify the Employee model by:

public class Employee

{

[Key]

public int EmployeeID { get; set; }

[Column("FirstName")]

[Required(ErrorMessage = "You must enter a {0}")]

[StringLength(30, ErrorMessage =

"The field {0} can contain maximun {1} and minimum {2} characters",

MinimumLength = 3)]

[Display(Name = "First Name")]

public string FirstName { get; set; }

[Required(ErrorMessage = "You must enter a {0}")]

[StringLength(30, ErrorMessage =

"The field {0} can contain maximun {1} and minimum {2} characters",

MinimumLength = 3)]

[Display(Name = "Last Name")]

public string LastName { get; set; }

[DisplayFormat(DataFormatString = "{0:C2}", ApplyFormatInEditMode = false)]

public decimal Salary { get; set; }

[Display(Name = "Bonus Percent")]

[DisplayFormat(DataFormatString = "{0:P2}", ApplyFormatInEditMode = false)]

[Range(0, 20, ErrorMessage = "The field {0} can take values between {1} and {2}")]

public float BonusPercent { get; set; }

[Display(Name = "Date Of Birth")]

[DataType(DataType.Date)]

[DisplayFormat(DataFormatString = "{0:yyyy-MM-dd}", ApplyFormatInEditMode = true)]

public DateTime DateOfBirth { get; set; }

[Display(Name = "Start Time")]

[DataType(DataType.Time)]

[DisplayFormat(DataFormatString = "{0:hh:mm}", ApplyFormatInEditMode = true)]

public DateTime StartTime { get; set; }

[DataType(DataType.EmailAddress)]

[Index("EMailIndex", IsUnique = true)]

public string EMail { get; set; }

[DataType(DataType.Url)]

public string URL { get; set; }

public int DocumentTypeID { get; set; }

public virtual DocumentType DocumentType { get; set; }

}

Sample to create a composite index:

[Index("CategoryDescriptionCompanyIdIndex", 1, IsUnique = true)]

public string Description { get; set; }

[Index("CategoryDescriptionCompanyIdIndex", 2, IsUnique = true)]

public int CompanyId { get; set; }

1. Modify the Document Type model by:

[Table("DocumentType")]

public class DocumentType

{

[Key]

public int DocumentTypeID { get; set; }

[Required(ErrorMessage = "You must enter a {0}")]

[Display(Name = "Document description")]

public string Description { get; set; }

public virtual ICollection<Employee> Employees { get; set; }

}

# MANY TO MANY RELATIONSHIP

1. Crete the new model Supplier:

public class Supplier

{

[Key]

public int SupplierID { get; set; }

public string Name { get; set; }

public string ContactFirstName { get; set; }

public string ContactLasttName { get; set; }

public string Phone { get; set; }

public string Address { get; set; }

public string EMail { get; set; }

public virtual ICollection<SupplierProduct> SuppliersProducts { get; set; }

}

1. Modify the Product model:

public class Product

{

[Key]

public int ProductID { get; set; }

public string Description { get; set; }

public decimal Price { get; set; }

public DateTime LastBuy { get; set; }

public float Stock { get; set; }

public string Remarks { get; set; }

public virtual ICollection<SupplierProduct> SuppliersProducts { get; set; }

}

1. Add the new model, Supplier Product:

public class SupplierProduct

{

[Key]

public int SupplierProductID { get; set; }

public int SupplierID { get; set; }

public int ProductID { get; set; }

public virtual Supplier Supplier { get; set; }

public virtual Product Product { get; set; }

}

1. Create the new controller for Suppliers, add some records and see the database.

# DISABLE CASCADE DELETING RULE

1. Add this method in the context database class:

protected override void OnModelCreating(DbModelBuilder modelBuilder)

{

modelBuilder.Conventions.Remove<OneToManyCascadeDeleteConvention>();

}

1. And modify the Delete method in the controller

public ActionResult DeleteConfirmed(int id)

{

DocumentType documentType = db.DocumentTypes.Find(id);

db.DocumentTypes.Remove(documentType);

try

{

db.SaveChanges();

}

catch { }

return RedirectToAction("Index");

}

**PENDING TO DOCUMENT THE ERROR MESSAGE**

# MENUS AND SUBMENUS

<div class="navbar-collapse collapse">

<ul class="nav navbar-nav" style="font-weight:bold;">

<li>@Html.ActionLink("Home", "Index", "Home")</li>

<li>@Html.ActionLink("About", "About", "Home")</li>

<li>@Html.ActionLink("Contact", "Contact", "Home")</li>

<li class="dropdown">

<a href="#" class="dropdown-toggle" data-toggle="dropdown">File<b class="caret"></b></a>

<ul class="dropdown-menu">

<li>@Html.ActionLink("Products", "Index", "Products")</li>

<li>@Html.ActionLink("Document Types", "Index", "DocumentTypes")</li>

<li>@Html.ActionLink("Employees", "Index", "Employees")</li>

</ul>

</li>

</ul>

@Html.Partial("\_LoginPartial")

</div>

# AJAX CALLS

1. Crete the new empty controller AJAX Concept:

public ActionResult Index()

{

return View();

}

public JsonResult JsonFactorial(int n)

{

if (!Request.IsAjaxRequest())

{

return null;

}

return new JsonResult

{

Data = new

{

Factorial = Factorial(n)

}

};

}

private double Factorial(int n)

{

double fac = 1;

for (int i = 2; i <= n; i++)

{

fac \*= i;

}

return fac;

}

1. Add the index view with the following lines:

@{

ViewBag.Title = "Home Page";

}

<h1>AJAX</h1>

@using (Html.BeginForm())

{

@Html.TextBox("txtN", 0)

<button id="btnCalculaFactorial">Calcular Factorial</button>

<div id="lblMessage"></div>

}

@section Scripts {

<script type="text/javascript">

$(function () {

$('#btnCalculaFactorial').on("click", function () {

$.ajax({

type: "POST",

url: '@Url.Action("JsonFactorial")',

data: { 'n': $('#txtN').val() },

datatype: 'json',

cache: false

( })

.success(function (data) {

$('#lblMessage').html(data.Factorial);

})

.error(function (xhr, ajaxOptions, thrownError) {

$('#lblMessage').html("There was an error");

});

return false;

});

});

</script>

}

1. Test it!

# WEB API & AJAX CALLS

1. Create the API controller to the required model.
2. Create the new action in any controller, with the following lines:

public ActionResult Students()

{

return View();

}

1. Add to the project a file like “loader.gif”
2. Add the view with the following lines:

@{

ViewBag.Title = "Students";

}

<h2>Students</h2>

@using (Html.BeginForm())

{

<button id="btnGetStudents">Get Students</button><br />

<button id="btnGetStudent">Get Student</button>

<input id="txtStudentID" type="text" value="1"/><br />

<button id="btnAddStudent">Add Student</button><br />

<button id="btnUpdateStudent">Update Student</button><br />

<button id="btnDeleteStudent">Delete Student</button>

<input id="txtStudentIDToDelete" type="text" value="1" /><br />

<div id="lblMensaje"></div>

}

@section Scripts

{

<script type="text/javascript">

$(function () {

$("#btnGetStudents").on("click", function () {

$.ajax({

type: "GET",

url: "/api/StudentsAPI",

datatype: "json",

cache: false,

})

.success(function (data) {

var outPut = "";

for (var i = 0; i < data.length; i++) {

outPut += "<hr/>";

outPut += "<b>Student ID : </b>" + data[i].StudentID + "<br/>";

outPut += "<b>Name : </b>" + data[i].Name + "<br/>";

outPut += "<b>Sure name : </b>" + data[i].SureName + "<br/>";

}

$("#lblMensaje").html(outPut);

})

.error(function (xhr, ajaxOptions, thrownError) {

$("#lblMensaje").html("Error: an error occured");

});

return false;

});

$("#btnGetStudent").on("click", function () {

var studentID = $("#txtStudentID").val();

$.ajax({

type: "GET",

url: "/api/StudentsAPI/" + studentID,

datatype: "json",

cache: false,

})

.success(function (data) {

if (data != null) {

var outPut = "";

outPut += "<hr/>";

outPut += "<b>Student ID : </b>" + data.StudentID + "<br/>";

outPut += "<b>Name : </b>" + data.Name + "<br/>";

outPut += "<b>Sure name : </b>" + data.SureName + "<br/>";

$("#lblMensaje").html(outPut);

} else {

$("#lblMensaje").html("No records found");

}

})

.error(function (xhr, ajaxOptions, thrownError) {

$("#lblMensaje").html("Error: an error occured");

});

return false;

});

$("#btnAddStudent").on("click", function () {

var student = { Name: "Valery", SureName: "Zuluaga" };

$.ajax({

type: "POST",

url: "/api/StudentsAPI/",

data: student,

datatype: "json",

cache: false,

})

.success(function (data) {

var outPut = "";

outPut += "<hr/><b>The new student is the following</b>";

outPut += "<b>Student ID : </b>" + data.StudentID + "<br/>";

outPut += "<b>Name : </b>" + data.Name + "<br/>";

outPut += "<b>Sure name : </b>" + data.SureName + "<br/>";

$("#lblMensaje").html(outPut);

})

.error(function (xhr, ajaxOptions, thrownError) {

$("#lblMensaje").html("Error: an error occured");

});

return false;

});

$("#btnUpdateStudent").on("click", function () {

var student = { StudentID: 1, Name: "Hector", SureName: "Lavoe" };

$.ajax({

type: "PUT",

url: "/api/StudentsAPI/1",

data: student,

datatype: "json",

cache: false,

})

.success(function (data) {

var outPut = "";

outPut += "<hr/><b>The update was complete</b>";

$("#lblMensaje").html(outPut);

})

.error(function (xhr, ajaxOptions, thrownError) {

$("#lblMensaje").html("Error: an error occured");

});

return false;

});

$("#btnDeleteStudent").on("click", function () {

var studentID = $("#txtStudentIDToDelete").val();

$.ajax({

type: "DELETE",

url: "/api/StudentsAPI/" + studentID,

datatype: "json",

cache: false,

})

.success(function (data) {

if (data != null) {

var outPut = "";

outPut += "<hr/><b>The student delete was</b>";

outPut += "<b>Student ID : </b>" + data.StudentID + "<br/>";

outPut += "<b>Name : </b>" + data.Name + "<br/>";

outPut += "<b>Sure name : </b>" + data.SureName + "<br/>";

$("#lblMensaje").html(outPut);

} else {

$("#lblMensaje").html("No records found");

}

})

.error(function (xhr, ajaxOptions, thrownError) {

$("#lblMensaje").html("Error: an error occured");

});

return false;

});

$(document).ajaxStart(function () {

$("#loading").show();

});

$(document).ajaxStop(function () {

$("#loading").hide();

});

});

</script>

}

<style type="text/css">

#loading {

display: none;

background-color: gray;

z-index: 999999;

position: absolute;

left: 0;

top: 0;

width: 100%;

height: 100%;

text-align:center;

padding-top: 300px;

filter: alpha(opacity=75);

-khtml-opacity: 0.75;

-moz-opacity: 0.75;

opacity: 0.75;

}

</style>

<div id="loading"><img src="~/images/loader.gif"/></div>

# SECURITY

## LOGIN

1. Create new user
2. Create the new application in Facebook developer
3. Edit the file Startup.Auth.cs in App\_Start folder, and put the application ID and application secret
4. Run the application and test it!!!

**Note**: for google see: <http://www.oauthforaspnet.com/providers/google/guides/aspnet-mvc5/>

## PERMISIONS

1. To demand access to the specific action, use [Authorize] in the action
2. To demand access to the whole controller, use [Authorize] in the controller
3. To allows anonymous in the specific action for a Authorize controller, uses: [AllowAnonymous]
4. To demand access to a specific user list, uses: [Authorize(Users = "jzuluaga55@gmail.com, jzuluaga55@hotmail.com")]
5. To demand access to a specific roll list, uses: [Authorize(Roles = "Admin")]

## CREATE ROLES AND SUPER USER

1. Add this line in the Global.asax:

CreateRolesAndSuperuser();

1. Implement the method:

private void CreateRolesAndSuperuser()

{

ApplicationDbContext db = new ApplicationDbContext();

CreateRoles(db);

CreateSuperuser(db);

AddPermisionsToSuperuser(db);

db.Dispose();

}

private void CreateRoles(ApplicationDbContext db)

{

var roleManager = new RoleManager<IdentityRole>(new RoleStore<IdentityRole>(db));

if (!roleManager.RoleExists("View"))

{

var result = roleManager.Create(new IdentityRole("View"));

}

if (!roleManager.RoleExists("Edit"))

{

var result = roleManager.Create(new IdentityRole("Edit"));

}

if (!roleManager.RoleExists("Create"))

{

var result = roleManager.Create(new IdentityRole("Create"));

}

if (!roleManager.RoleExists("Delete"))

{

var result = roleManager.Create(new IdentityRole("Delete"));

}

}

private void CreateSuperuser(ApplicationDbContext db)

{

var userManager = new UserManager<ApplicationUser>(new UserStore<ApplicationUser>(db));

var user = userManager.FindByName("jzuluaga55@hotmail.com");

if (user == null)

{

user = new ApplicationUser();

user.UserName = "jzuluaga55@hotmail.com";

user.Email = "jzuluaga55@hotmail.com";

var result = userManager.Create(user, "Zulu123.");

}

}

private void AddPermisionsToSuperuser(ApplicationDbContext db)

{

var userManager = new UserManager<ApplicationUser>(new UserStore<ApplicationUser>(db));

var roleManager = new RoleManager<IdentityRole>(new RoleStore<IdentityRole>(db));

var user = userManager.FindByName("jzuluaga55@hotmail.com");

if (!userManager.IsInRole(user.Id, "View"))

{

userManager.AddToRole(user.Id, "View");

}

if (!userManager.IsInRole(user.Id, "Edit"))

{

userManager.AddToRole(user.Id, "Edit");

}

if (!userManager.IsInRole(user.Id, "Create"))

{

userManager.AddToRole(user.Id, "Create");

}

if (!userManager.IsInRole(user.Id, "Delete"))

{

userManager.AddToRole(user.Id, "Delete");

}

}

# MASTER DETAIL

## MODELS

namespace Z\_Market.Models

{

public class Order

{

[Key]

public int OrderID { get; set; }

[Required]

public int CustomerID { get; set; }

[Required]

[DataType(DataType.DateTime)]

public DateTime DateOrder { get; set; }

[Required]

public OrderStatus OrderStatus { get; set; }

public virtual Customer Customer { get; set; }

public virtual ICollection<OrderDetails> Details { get; set; }

}

}

namespace Z\_Market.Models

{

public class OrderDetails

{

[Key]

public int OrderDetailsID { get; set; }

public int OrderID { get; set; }

public int ProductID { get; set; }

[Display(Name = "Product Description")]

public string Description { get; set; }

[DataType(DataType.Currency)]

[DisplayFormat(DataFormatString = "{0:C2}", ApplyFormatInEditMode = false)]

public decimal Price { get; set; }

[DataType(DataType.Currency)]

[DisplayFormat(DataFormatString = "{0:N2}", ApplyFormatInEditMode = false)]

public float Quantity { get; set; }

public virtual Order Order { get; set; }

public virtual Product Product { get; set; }

}

}

In the context:

public System.Data.Entity.DbSet<Z\_Market.Models.Order> Orders { get; set; }

public System.Data.Entity.DbSet<Z\_Market.Models.OrderDetails> OrdersDetails { get; set; }

In the Model Views:

namespace Z\_Market.ModelViews

{

public class OrderView

{

public Customer Customer { get; set; }

public ProductOrder Product { get; set; }

public List<ProductOrder> Products { get; set; }

}

}

namespace Z\_Market.Models

{

public class Customer

{

[Key]

[Required(ErrorMessage = "You must enter the field {0}")]

[Range(1, 999999, ErrorMessage = "You must enter a valid value in the field {0}")]

public int CustomerID { get; set; }

[StringLength(30, ErrorMessage = "The field {0} must contain between {2} and {1} characters", MinimumLength = 3)]

[Required(ErrorMessage = "You must enter the field {0}")]

[Display(Name = "First Name")]

public string FirstName { get; set; }

[StringLength(30, ErrorMessage = "The field {0} must contain between {2} and {1} characters", MinimumLength = 3)]

[Required(ErrorMessage = "You must enter the field {0}")]

[Display(Name = "Last Name")]

public string LastName { get; set; }

[StringLength(20, ErrorMessage = "The field {0} must contain between {2} and {1} characters", MinimumLength = 5)]

[Required(ErrorMessage = "You must enter the field {0}")]

public string Document { get; set; }

[Display(Name = "Document Type")]

public int DocumentTypeID { get; set; }

[DataType(DataType.PhoneNumber)]

[StringLength(30, ErrorMessage = "The field {0} must contain between {2} and {1} characters", MinimumLength = 3)]

[Required(ErrorMessage = "You must enter the field {0}")]

public string Phone { get; set; }

[StringLength(30, ErrorMessage = "The field {0} must contain between {2} and {1} characters", MinimumLength = 3)]

[Required(ErrorMessage = "You must enter the field {0}")]

public string Address { get; set; }

[DataType(DataType.EmailAddress)]

public string EMail { get; set; }

[NotMapped]

public string FullName { get { return string.Format("{0} {1}", FirstName, LastName) ;} }

public virtual DocumentType DocumentType { get; set; }

public virtual ICollection<Order> Orders { get; set; }

}

}

namespace Z\_Market.Models

{

public class ProductOrder: Product

{

[DataType(DataType.Currency)]

[Required(ErrorMessage = "You must enter the field {0}")]

[DisplayFormat(DataFormatString = "{0:N2}", ApplyFormatInEditMode = false)]

public float Quantity { get; set; }

[DataType(DataType.Currency)]

[DisplayFormat(DataFormatString = "{0:C2}", ApplyFormatInEditMode = false)]

public decimal Value { get { return Price \* (decimal)Quantity; } }

}

}

namespace Z\_Market.Models

{

public class Product

{

[Key]

[Required(ErrorMessage = "You must enter the field {0}")]

[Range(1, 999999, ErrorMessage = "You must enter a valid value in the field {0}")]

public int ProductID { get; set; }

[StringLength(30, ErrorMessage = "The field {0} must contain between {2} and {1} characters", MinimumLength = 3)]

[Required(ErrorMessage = "You must enter the field {0}")]

[Display(Name = "Product Description")]

public string Description { get; set; }

[DataType(DataType.Currency)]

[DisplayFormat(DataFormatString = "{0:C2}", ApplyFormatInEditMode = false)]

[Required(ErrorMessage = "You must enter the field {0}")]

public decimal Price { get; set; }

[DataType(DataType.Date)]

[DisplayFormat(DataFormatString = "{0:dd/MM/yyyy}", ApplyFormatInEditMode = true)]

[Display(Name = "Last Buy")]

public DateTime LastBuy { get; set; }

[DataType(DataType.Currency)]

[DisplayFormat(DataFormatString = "{0:N2}", ApplyFormatInEditMode = false)]

public float Stock { get; set; }

[DataType(DataType.MultilineText)]

public string Remarks { get; set; }

public virtual ICollection<SupplierProduct> SupplierProducts { get; set; }

}

}

## CONTROLLERS

namespace Z\_Market.Controllers

{

public class OrdersController : Controller

{

Z\_MarketContext db = new Z\_MarketContext();

public ActionResult NewOrder()

{

var orderView = new OrderView();

orderView.Customer = new Customer();

orderView.Products = new List<ProductOrder>();

Session["orderView"] = orderView;

var customers = db.Customers.ToList();

customers.Add(new Customer { CustomerID = 0, FirstName = "[Select a customer...]" });

ViewBag.CustomerID = new SelectList(

customers.OrderBy(c => c.FullName),

"CustomerID", "FullName", orderView.Customer.CustomerID);

return View(orderView);

}

[HttpPost]

public ActionResult NewOrder(OrderView orderView)

{

orderView = Session["orderView"] as OrderView;

int customerID = int.Parse(Request["CustomerID"]);

if(customerID == 0)

{

ViewBag.Error = "You must select a customer";

var customers = db.Customers.ToList();

customers.Add(new Customer { CustomerID = 0, FirstName = "[Select a customer...]" });

ViewBag.CustomerID = new SelectList(

customers.OrderBy(c => c.FullName),

"CustomerID", "FullName", orderView.Customer.CustomerID);

return View(orderView);

}

var customer = db.Customers.Find(customerID);

if (customer == null)

{

ViewBag.Error = "The customer does not exit";

var customers = db.Customers.ToList();

customers.Add(new Customer { CustomerID = 0, FirstName = "[Select a customer...]" });

ViewBag.CustomerID = new SelectList(

customers.OrderBy(c => c.FullName),

"CustomerID", "FullName", orderView.Customer.CustomerID);

return View(orderView);

}

orderView.Customer.CustomerID = customerID;

if (orderView.Products.Count == 0)

{

ViewBag.Error = "You must enter details";

var customers = db.Customers.ToList();

customers.Add(new Customer { CustomerID = 0, FirstName = "[Select a customer...]" });

ViewBag.CustomerID = new SelectList(

customers.OrderBy(c => c.FullName),

"CustomerID", "FullName", orderView.Customer.CustomerID);

return View(orderView);

}

var order = new Order

{

CustomerID = customerID,

DateOrder = DateTime.Now,

OrderStatus = OrderStatus.Created

};

int orderID = 0;

using (var transaction = db.Database.BeginTransaction())

{

try

{

db.Orders.Add(order);

db.SaveChanges();

orderID = db.Orders.Select(o => o.OrderID).Max();

foreach (var item in orderView.Products)

{

var orderDetail = new OrderDetails

{

Description = item.Description,

OrderID = orderID,

Price = item.Price,

ProductID = item.ProductID,

Quantity = item.Quantity

};

db.OrdersDetails.Add(orderDetail);

}

db.SaveChanges();

transaction.Commit();

}

catch (Exception ex)

{

transaction.Rollback();

ViewBag.Error = "Error:" + ex.Message;

var cstomers = db.Customers.ToList();

cstomers.Add(new Customer { CustomerID = 0, FirstName = "[Select a customer...]" });

ViewBag.CustomerID = new SelectList(

cstomers.OrderBy(c => c.FullName),

"CustomerID", "FullName", orderView.Customer.CustomerID);

return View(orderView);

}

}

ViewBag.Message = string.Format("Order: {0} saved sucessfully", orderID);

orderView.Customer = new Customer();

orderView.Products = new List<ProductOrder>();

var cstomrs = db.Customers.ToList();

cstomrs.Add(new Customer { CustomerID = 0, FirstName = "[Select a customer...]" });

ViewBag.CustomerID = new SelectList(

cstomrs.OrderBy(c => c.FullName),

"CustomerID", "FullName", orderView.Customer.CustomerID);

return View(orderView);

}

public ActionResult AddProduct()

{

var productOrder = new ProductOrder();

var products = db.Products.ToList();

products.Add(new Product { ProductID = 0, Description = "[Select a product...]" });

ViewBag.ProductID = new SelectList(

products.OrderBy(p => p.Description),

"ProductID", "Description", productOrder.ProductID);

return View();

}

[HttpPost]

public ActionResult AddProduct(ProductOrder productOrder)

{

var orderView = Session["orderView"] as OrderView;

var productID = int.Parse(Request["ProductID"]);

if(productID == 0)

{

var products = db.Products.ToList();

products.Add(new Product { ProductID = 0, Description = "[Select a product...]" });

ViewBag.ProductID = new SelectList(

products.OrderBy(p => p.Description),

"ProductID", "Description", productOrder.ProductID);

return View(productOrder);

}

var product = db.Products.Find(productID);

if (product == null)

{

var products = db.Products.ToList();

products.Add(new Product { ProductID = 0, Description = "[Select a product...]" });

ViewBag.ProductID = new SelectList(

products.OrderBy(p => p.Description),

"ProductID", "Description", productOrder.ProductID);

return View(productOrder);

}

float quantity = 0;

if (!float.TryParse(Request["Quantity"], out quantity) || quantity <= 0)

{

var products = db.Products.ToList();

products.Add(new Product { ProductID = 0, Description = "[Select a product...]" });

ViewBag.ProductID = new SelectList(

products.OrderBy(p => p.Description),

"ProductID", "Description", productOrder.ProductID);

return View(productOrder);

}

productOrder = orderView.Products.Find(p => p.ProductID == productID);

if (productOrder == null)

{

productOrder = new ProductOrder

{

Description = product.Description,

LastBuy = product.LastBuy,

Price = product.Price,

ProductID = product.ProductID,

Quantity = float.Parse(Request["Quantity"]),

Remarks = product.Remarks,

Stock = product.Stock

};

orderView.Products.Add(productOrder);

}

else

{

productOrder.Quantity += quantity;

}

var customers = db.Customers.ToList();

customers.Add(new Customer { CustomerID = 0, FirstName = "[Select a customer...]" });

ViewBag.CustomerID = new SelectList(

customers.OrderBy(c => c.FullName),

"CustomerID", "FullName", orderView.Customer.CustomerID);

return View("NewOrder", orderView);

}

}

}

## VIEWS

@model Z\_Market.ModelViews.OrderView

@{

ViewBag.Title = "NewOrder";

}

<h2>New Order</h2>

<h3>@ViewBag.Error</h3>

<h3>@ViewBag.Message</h3>

@using (Html.BeginForm("NewOrder", "Orders", FormMethod.Post))

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

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

<div class="form-group">

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

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

@Html.DropDownList("CustomerID", null, htmlAttributes: new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.Customer.CustomerID, "", new { @class = "text-danger" })

</div>

</div>

@Html.ActionLink("Add Product", "AddProduct", new { }, new { @class = "btn btn-default" })

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

</div>

}

<h3>Details</h3>

<table class="table">

<tr>

<th>

@Html.DisplayNameFor(model => model.Product.Description)

</th>

<th>

@Html.DisplayNameFor(model => model.Product.Price)

</th>

<th>

@Html.DisplayNameFor(model => model.Product.Quantity)

</th>

<th>

@Html.DisplayNameFor(model => model.Product.Value)

</th>

<th></th>

</tr>

@for (int i = 0; i < Model.Products.Count; i++)

{

<tr>

<td>

@Html.DisplayFor(modelItem => Model.Products[i].Description)

</td>

<td>

@Html.DisplayFor(modelItem => Model.Products[i].Price)

</td>

<td>

@Html.DisplayFor(modelItem => Model.Products[i].Quantity)

</td>

<td>

@Html.DisplayFor(modelItem => Model.Products[i].Value)

</td>

<td>

@Html.ActionLink("Edit", "Edit", new { id = Model.Products[i].ProductID }) |

@Html.ActionLink("Delete", "Delete", new { id = Model.Products[i].ProductID })

</td>

</tr>

}

</table>

@model Z\_Market.Models.ProductOrder

@{

ViewBag.Title = "AddProduct";

}

<h2>Add Product</h2>

<h3>@ViewBag.Error</h3>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

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

<div class="form-group">

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

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

@Html.DropDownList("ProductID", null, htmlAttributes: new { @class = "form-control" })

@Html.ValidationMessageFor(model => model.ProductID, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

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

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

@Html.EditorFor(model => model.Quantity, new { htmlAttributes = new { @class = "form-control" } })

@Html.ValidationMessageFor(model => model.Quantity, "", new { @class = "text-danger" })

</div>

</div>

<div class="form-group">

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

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

</div>

</div>

</div>

}

# PUBLISH THE APP

1. Backup and compress Data Base.
2. Make this changes in the Web Config
   1. The connection strings:

<remove name="DefaultConnection"/>

<add name="DefaultConnection"

connectionString="Data Source=10.61.44.196;Initial Catalog=Z-Market2;Persist Security Info=True;User ID=zuluadmin;Password=zuluadmin2015"

providerName="System.Data.SqlClient" />

<remove name="Z\_MarketContext"/>

<add name="Z\_MarketContext"

connectionString="Data Source=10.61.44.196;Initial Catalog=Z-Market2;Persist Security Info=True;User ID=zuluadmin;Password=zuluadmin2015"

providerName="System.Data.SqlClient" />

* 1. In the run time:

<dependentAssembly>

<assemblyIdentity name="EntityFramework" publicKeyToken="b77a5c561934e089" culture="neutral" />

<bindingRedirect oldVersion="0.0.0.0-5.0.0.0" newVersion="6.0.0.0" />

</dependentAssembly>

* 1. Comment the entity framework in configSections:

<configSections>

<!--<section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=6.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false" />-->

<!-- For more information on Entity Framework configuration, visit http://go.microsoft.com/fwlink/?LinkID=237468 -->

</configSections>

<!-- Azure -->

<!--<add name="DefaultConnection"

connectionString="Data Source=zuluserver.database.windows.net;Initial Catalog=ZuluDB;Persist Security Info=True;User ID=Zulu;Password=Roger1974"

providerName="System.Data.SqlClient" />-->

<!-- Local -->

<!--<add name="DefaultConnection"

connectionString="Data Source=.;Initial Catalog=ZuluDB;Integrated Security=True"

providerName="System.Data.SqlClient" />-->

<!-- Negox -->

<add name="DefaultConnection"

connectionString="Data Source=10.61.44.196;Initial Catalog=zulusof1\_zulubd;Persist Security Info=True;User ID=zulusof1\_zulu;Password=Roger1974."

providerName="System.Data.SqlClient" />

# UPLOAD AND SHOW IMAGES

1. Create a new model called UserView, with all the attributes and change the attribute string photo by:

public HttpPostedFileBase Photo { get; set; }

1. Create the folder Photos into Content folder
2. Change the view and replace the photo section by:

<div class="form-group">

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

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

<span class="btn btn-default btn-file">

@Html.TextBoxFor(modelo => modelo.LogoFile, new { type = "file" })

</span>

</div>

</div>

1. Don’t forget change this line in the view by:

@using (Html.BeginForm("Create", "Users", FormMethod.Post, new { enctype = "multipart/form-data" }))

1. Create the helper class with the method to upload files:

public static string UploadPhoto(HttpPostedFileBase file, string folder)

{

string path = string.Empty;

string pic = string.Empty;

if (file != null)

{

pic = Path.GetFileName(file.FileName);

path = Path.Combine(HttpContext.Current.Server.MapPath(folder), pic);

file.SaveAs(path);

}

return pic;

}

1. Change the create and edit method to call the upload photo:

var pic = string.Empty;

var folder = "~/Content/Logos";

if (view.LogoFile != null)

{

pic = FilesHelper.UploadPhoto(view.LogoFile, folder);

pic = string.Format("{0}/{1}", folder, pic);

}

1. Change the index view for show the image:

@if (!string.IsNullOrEmpty(item.Logo))

{

<img src="@Url.Content(item.Logo)" alt="Image" style="width:100px;height:150px;max-width: 100%; height: auto;" />

}

# CREATE ROLE, USER AND ASSING ROLE TO USER

1. After save changes in create user, add the line:

this.CreateASPUser(userView);

1. Add this method:

private void CreateASPUser(UserView userView)

{

// User management

var userContext = new ApplicationDbContext();

var userManager = new UserManager<ApplicationUser>(new UserStore<ApplicationUser>(userContext));

var roleManager = new RoleManager<IdentityRole>(new RoleStore<IdentityRole>(userContext));

// Create User role

string roleName = "User";

// Check to see if Role Exists, if not create it

if (!roleManager.RoleExists(roleName))

{

roleManager.Create(new IdentityRole(roleName));

}

// Create the ASP NET User

var userASP = new ApplicationUser

{

UserName = userView.UserName,

Email = userView.UserName,

PhoneNumber = userView.Phone,

};

userManager.Create(userASP, userASP.UserName);

// Add user to role

userASP = userManager.FindByName(userView.UserName);

userManager.AddToRole(userASP.Id, "User");

}

# MASTER DETAIL FORM

1. Creates the Group Member model, with their relations:

public class GroupMember

{

[Key]

public int GroupMemberId { get; set; }

public int GroupId { get; set; }

public int UserId { get; set; }

public virtual Group Group { get; set; }

public virtual User User { get; set; }

}

1. Add the relation in group model:

public virtual ICollection<GroupMember> GroupMembers { get; set; }

1. Add the relation in user model:

public virtual ICollection<GroupMember> GroupMembers { get; set; }

1. In the context add the respective DB Set to the group member model:

public DbSet<GroupMember> GroupMembers { get; set; }

1. Create the Group View class, this class will be the main class to build the master detail form:

public class GroupView

{

/// <summary>

/// Gets or sets the group id

/// </summary>

public int GroupId { get; set; }

/// <summary>

/// Gets or sets the group description

/// </summary>

[Required(ErrorMessage = "The field {0} is required")]

[Display(Name = "Group description")]

public string Description { get; set; }

/// <summary>

/// Gets or sets the group members

/// </summary>

public List<GroupMember> GroupMembers { get; set; }

}

1. Update the details action in group controller:

public ActionResult Details(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

var group = db.Groups.Find(id);

if (group == null)

{

return HttpNotFound();

}

var groupView = new GroupView

{

Description = group.Description,

GroupId = group.GroupId,

GroupMembers = group.GroupMembers.ToList(),

};

return View(groupView);

}

1. Change the view:

@model Democracy.Models.GroupView

@{

ViewBag.Title = "Details";

}

<h2>Details</h2>

<div>

<h4>Group</h4>

<hr />

<dl class="dl-horizontal">

<dt>

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

</dt>

<dd>

@Html.DisplayFor(model => model.Description)

</dd>

</dl>

</div>

<p>

@Html.ActionLink("Edit", "Edit", new { id = Model.GroupId }, new { @class = "btn btn-primary" })

@Html.ActionLink("Add Member", "AddMember", new { id = Model.GroupId }, new { @class = "btn btn-default" })

@Html.ActionLink("Back to List", "Index", new { }, new { @class = "btn btn-success" })

</p>

<h3>Members</h3>

@if (Model.GroupMembers.Count > 0)

{

<table class="table">

<tr>

<th>

@Html.DisplayNameFor(model => Model.GroupMembers[0].User.FirstName)

</th>

<th>

@Html.DisplayNameFor(model => Model.GroupMembers[0].User.LastName)

</th>

<th>

@Html.DisplayNameFor(model => Model.GroupMembers[0].User.UserName)

</th>

<th>

@Html.DisplayNameFor(model => Model.GroupMembers[0].User.Grade)

</th>

<th>

@Html.DisplayNameFor(model => Model.GroupMembers[0].User.Group)

</th>

<th></th>

</tr>

@for (int i = 0; i < Model.GroupMembers.Count; i++)

{

<tr>

<td>

@Html.DisplayFor(modelItem => Model.GroupMembers[i].User.FirstName)

</td>

<td>

@Html.DisplayFor(modelItem => Model.GroupMembers[i].User.LastName)

</td>

<td>

@Html.DisplayFor(modelItem => Model.GroupMembers[i].User.UserName)

</td>

<td>

@Html.DisplayFor(modelItem => Model.GroupMembers[i].User.Grade)

</td>

<td>

@Html.DisplayFor(modelItem => Model.GroupMembers[i].User.Group)

</td>

<td>

@Html.ActionLink("Delete", "DeleteMember",

new { id = Model.GroupMembers[i].GroupMemberId },

new { onclick = "return confirm('Are you want to delete this record?');", @class = "btn btn-danger" })

</td>

</tr>

}

</table>

}

else

{

<h4>No members yet</h4>

}

1. Add the read only property in user model:

[Display(Name = "Full name")]

public string FullName { get { return string.Format("{0} {1}", this.FirstName, this.LastName); } }

1. Add the action Add Member in group controller:

[HttpGet]

public ActionResult AddMember(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

var group = db.Groups.Find(id);

if (group == null)

{

return HttpNotFound();

}

ViewBag.UserId = new SelectList(db.Users.OrderBy(u => u.FirstName).ThenBy(u => u.LastName), "UserId", "FullName");

ViewBag.GroupName = group.Description;

var groupMember = new GroupMember

{

GroupId = group.GroupId,

};

return View(groupMember);

}

[HttpPost]

public ActionResult AddMember(GroupMember newGroupMember)

{

if (!ModelState.IsValid)

{

return View(newGroupMember);

}

var groupMember = db.GroupMembers

.Where(gm => gm.GroupId == newGroupMember.GroupId &&

gm.UserId == newGroupMember.UserId)

.FirstOrDefault();

if (groupMember != null)

{

var group = db.Groups.Find(newGroupMember.GroupId);

ViewBag.GroupName = group.Description;

ViewBag.Error = "User already belongs to group";

ViewBag.UserId = new SelectList(db.Users.OrderBy(u => u.FirstName).ThenBy(u => u.LastName), "UserId", "FullName");

return View(newGroupMember);

}

db.GroupMembers.Add(newGroupMember);

db.SaveChanges();

return RedirectToAction(string.Format("Details/{0}", newGroupMember.GroupId));

}

1. Add the view Add Member:

@model Democracy.Models.GroupMember

@{

ViewBag.Title = "AddMember";

}

<h2>Add Member</h2>

<h3>@ViewBag.Error</h3>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<h4>To group: @ViewBag.GroupName</h4>

<hr />

@Html.ValidationSummary(true)

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

<div class="form-group">

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

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

@Html.DropDownList("UserId", String.Empty)

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

</div>

</div>

<div class="form-group">

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

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

</div>

</div>

</div>

}

<div>

@Html.ActionLink("Back to List", "Index", new { }, new { @class = "btn btn-success" })

</div>

@section Scripts {

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

}

1. Add the action Delete Member:

[HttpGet]

public ActionResult DeleteMember(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

var groupMember = db.GroupMembers.Find(id);

if (groupMember == null)

{

return HttpNotFound();

}

db.GroupMembers.Remove(groupMember);

db.SaveChanges();

return RedirectToAction(string.Format("Details/{0}", groupMember.GroupId));

}

1. At the last, modify the index view, to show the count members per group:

<table class="table">

<tr>

<th>

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

</th>

<th>

Members Quantity

</th>

<th></th>

</tr>

@foreach (var item in Model) {

<tr>

<td>

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

</td>

<td>

@Html.DisplayFor(modelItem => item.GroupMembers.Count)

</td>

<td>

@Html.ActionLink("Edit", "Edit", new { id = item.GroupId }, new { @class = "btn btn-info" })

@Html.ActionLink("Details", "Details", new { id = item.GroupId }, new { @class = "btn btn-warning" })

@Html.ActionLink("Delete", "Delete", new { id = item.GroupId }, new { @class = "btn btn-danger" })

</td>

</tr>

}

# ENABLE WEB TRACE

1. Add this line in Web.config in <system.web> section:

<trace enabled="true" requestLimit="50" localOnly="false"/>

1. Try catch the possible segment error (example):

try

{

con.Open();

var cmd = new SqlCommand(sql, con);

var adp = new SqlDataAdapter(cmd);

adp.Fill(dt);

var rpt = new ReportClass();

rpt.FileName = Server.MapPath("~/Reports/Users.rpt");

rpt.Load();

rpt.SetDataSource(dt);

return rpt;

}

catch (Exception ex)

{

ex.ToString();

HttpContext.Trace.Warn("ERROR: " + ex.ToString());

}

1. Now, shoot the error and write this in URL command:

<http://zulu-software.com/Democracy/Trace.axd>

And analyze the error

# CRYSTAL REPORTS IN APP PUBLISHED

When the application is published in WEB, does not work ☹. You need to change this properties in the RPT file, and the report magically works!



# CASCADE DROPDOWNS

1. Create the method to retrieve the data in the controller:

public JsonResult GetMunicipalities(int departmentId)

{

db.Configuration.ProxyCreationEnabled = false;

var municipalities = db.Municipalities.Where(m => m.DepartmentId == departmentId);

return Json(municipalities);

}

1. Add this section in Create and Edit view:

@section Scripts {

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

<script type="text/javascript">

$(document).ready(function () {

$("#DepartmentId").change(function () {

$("#MunicipalityId").empty();

$.ajax({

type: 'POST',

url: '@Url.Action("GetMunicipalities")',

dataType: 'json',

data: { departmentId: $("#DepartmentId").val() },

success: function (municipalities) {

$.each(municipalities, function (i, municipality) {

$("#MunicipalityId").append('<option value="'

+ municipality.MunicipalityId + '">'

+ municipality.Name + '</option>');

});

},

error: function (ex) {

alert('Failed to retrieve municipalities.' + ex);

}

});

return false;

})

});

</script>

}

1. To get the initial values in correct way, modify the create action:

public ActionResult Create()

{

ViewBag.DepartmentId = new SelectList(db.Departaments,

"DepartmentId", "Name");

ViewBag.MunicipalityId = new SelectList(db.Municipalities

.Where(m => m.DepartmentId == db.Departaments.FirstOrDefault().DepartmentId),

"MunicipalityId", "Name");

ViewBag.DocumentTypeId = new SelectList(db.DocumentTypes,

"DocumentTypeId", "Description");

return View();

}

And replace in the Edit views, the way to fill up the dropdown lists:

ViewBag.DepartmentId = new SelectList(db.Departaments,

"DepartmentId", "Name",

taxPaer.DepartmentId);

ViewBag.MunicipalityId = new SelectList(db.Municipalities

.Where(m => m.DepartmentId == taxPaer.DepartmentId),

"MunicipalityId", "Name",

taxPaer.MunicipalityId);

ViewBag.DocumentTypeId = new SelectList(db.DocumentTypes,

"DocumentTypeId", "Description",

taxPaer.DocumentTypeId);

# RECOVER & EMAIL PASSWORD

1. Add some values in the web.config file:

<appSettings>

<add key="webpages:Version" value="3.0.0.0" />

<add key="webpages:Enabled" value="false" />

<add key="ClientValidationEnabled" value="true" />

<add key="UnobtrusiveJavaScriptEnabled" value="true" />

<add key="AdminUser" value="taxesapplication@gmail.com"/>

<add key="AdminPassWord" value="Taxes123."/>

<add key="SMTPName" value="smtp.gmail.com"/>

<add key="SMTPPort" value="587"/>

</appSettings>

1. Modify the login view:

<p>

@Html.ActionLink("Forgot your password?", "ForgotPassword")

</p>

1. Add the method SendMail in Utilities class:

public static async Task SendMail(string to, string subject, string body)

{

var message = new MailMessage();

message.To.Add(new MailAddress(to));

message.From = new MailAddress(WebConfigurationManager.AppSettings["AdminUser"]);

message.Subject = subject;

message.Body = body;

message.IsBodyHtml = true;

using (var smtp = new SmtpClient())

{

var credential = new NetworkCredential

{

UserName = WebConfigurationManager.AppSettings["AdminUser"],

Password = WebConfigurationManager.AppSettings["AdminPassWord"]

};

smtp.Credentials = credential;

smtp.Host = WebConfigurationManager.AppSettings["SMTPName"];

smtp.Port = int.Parse(WebConfigurationManager.AppSettings["SMTPPort"]);

smtp.EnableSsl = true;

await smtp.SendMailAsync(message);

}

}

1. Add the method PasswordRecovery in Utilities class:

public static async Task PasswordRecovery(string email)

{

var userManager = new UserManager<ApplicationUser>(new UserStore<ApplicationUser>(userContext));

var userASP = userManager.FindByEmail(email);

if (userASP == null)

{

return;

}

var user = db.TaxPaers.Where(tp => tp.UserName == email).FirstOrDefault();

if (user == null)

{

return;

}

var random = new Random();

var newPassword = string.Format("{0}{1}{2:04}\*", user.FirstName.ToUpper().Substring(0, 1), user.LastName.ToLower(), random.Next(9999));

userManager.RemovePassword(userASP.Id);

userManager.AddPassword(userASP.Id, newPassword);

var subject = "Taxes Password Recovery";

var body = string.Format(@"

<h1>Taxes Password Recovery</h1>

<p>Yor new password is: <strong>{0}</strong></p>

<p>Please change it for one, that you remember easyly",

newPassword);

await SendMail(email, subject, body);

}

1. Modify the ForgotPassword action by:

[HttpPost]

[AllowAnonymous]

[ValidateAntiForgeryToken]

public async Task<ActionResult> ForgotPassword(ForgotPasswordViewModel model)

{

if (ModelState.IsValid)

{

var user = await UserManager.FindByNameAsync(model.Email);

if (user != null)

{

await Utilities.PasswordRecovery(model.Email);

return RedirectToAction("ForgotPasswordConfirmation", "Account");

}

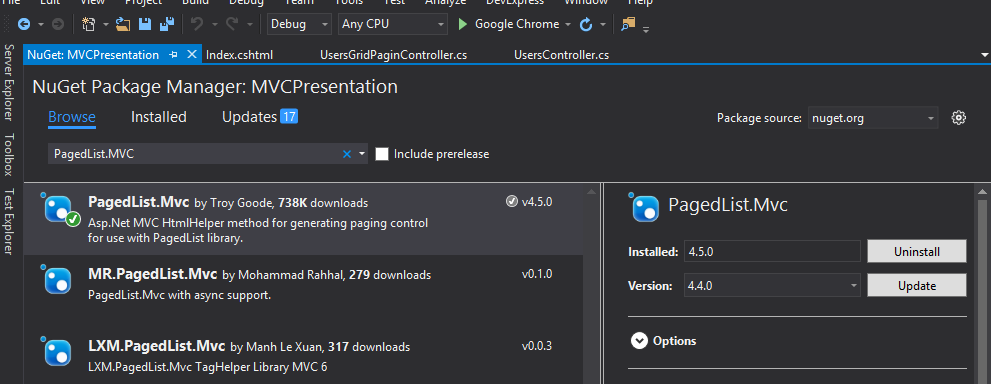
}

return View(model);

}

# PAGING

1. Add the nugget PagedList.MVC



1. Change the Index in TaxPaerController by (don’t forget add using PagedList;):

[Authorize(Roles = "Admin")]

public ActionResult Index(int? page = null)

{

page = (page ?? 1);

return View(db.TaxPaers

.OrderBy(tp => tp.FirstName)

.ThenBy(tp => tp.LastName)

.ToPagedList((int)page, 5));

}

1. Modify the view Index by:

@model PagedList.IPagedList<Taxes.Models.TaxPaer>

@using PagedList.Mvc;

@{

ViewBag.Title = "Tax Paers";

}

<h2>Tax Paers</h2>

<p>

@Html.ActionLink("Create New", "Create", new { }, new { @class = "btn btn-primary" })

</p>

<table class="table">

<tr>

<th>

@Html.DisplayNameFor(model => model.FirstOrDefault().DocumentType.Description)

</th>

<th>

@Html.DisplayNameFor(model => model.FirstOrDefault().FirstName)

</th>

<th>

@Html.DisplayNameFor(model => model.FirstOrDefault().LastName)

</th>

<th>

@Html.DisplayNameFor(model => model.FirstOrDefault().UserName)

</th>

<th>

@Html.DisplayNameFor(model => model.FirstOrDefault().Phone)

</th>

<th>

@Html.DisplayNameFor(model => model.FirstOrDefault().Document)

</th>

<th>

# Properties

</th>

<th></th>

</tr>

@foreach (var item in Model) {

<tr>

<td>

@Html.DisplayFor(modelItem => item.DocumentType.Description)

</td>

<td>

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

</td>

<td>

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

</td>

<td>

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

</td>

<td>

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

</td>

<td>

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

</td>

<td>

@Html.DisplayFor(modelItem => item.Properties.Count)

</td>

<td>

@Html.ActionLink("Edit", "Edit", new { id = item.TaxPaerId }, new { @class = "btn btn-warning" })

@Html.ActionLink("Details", "Details", new { id=item.TaxPaerId }, new { @class = "btn btn-info" })

@Html.ActionLink("Delete", "Delete", new { id=item.TaxPaerId }, new { @class = "btn btn-danger" })

</td>

</tr>

}

</table>

Página: @(Model.PageCount < Model.PageNumber ? 0 : Model.PageNumber), de: @Model.PageCount

@Html.PagedListPager(Model, page => Url.Action("Index", new

{

page,

sortOrder =

ViewBag.CurrentSort,

currentFilter = ViewBag.CurrentFilter

}))

# FILTERS IN VIEW

1. Create the model MunicipalitiesView

public class MunicipalitiesView

{

public string Department { get; set; }

public string Municipality { get; set; }

public List<Municipality> Municipalities { get; set; }

}

1. Modify the action Index:

public ActionResult Index()

{

var municipalities = db.Municipalities

.Include(m => m.Departament)

.OrderBy(m => m.Departament.Name)

.ThenBy(m => m.Name)

.ToList();

var view = new MunicipalitiesView

{

Municipalities = municipalities,

};

return View(view);

}

1. Modify the index view:

@model Taxes.Models.MunicipalitiesView

@{

ViewBag.Title = "Municipalities";

}

<h2>Municipalities</h2>

@using (Html.BeginForm())

{

<table>

<tr>

<td>

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

</td>

<td>

@Html.EditorFor(model => model.Department, new { htmlAttributes = new { @class = "form-control" } })

</td>

<td>

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

</td>

<td>

@Html.EditorFor(model => model.Municipality, new { htmlAttributes = new { @class = "form-control" } })

</td>

<td>

<input type="submit" value="Filter" class="btn btn-info" />

@Html.ActionLink("Create New", "Create", new { }, new { @class = "btn btn-primary" })

</td>

</tr>

</table>

}

<br />

<table class="table">

<tr>

<th>

@Html.DisplayNameFor(model => model.Municipalities.FirstOrDefault().Departament.Name)

</th>

<th>

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

</th>

<th></th>

</tr>

@foreach (var item in Model.Municipalities) {

<tr>

<td>

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

</td>

<td>

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

</td>

<td>

@Html.ActionLink("Edit", "Edit", new { id = item.MunicipalityId }, new { @class = "btn btn-warning" })

@Html.ActionLink("Details", "Details", new { id=item.MunicipalityId }, new { @class = "btn btn-info" })

@Html.ActionLink("Delete", "Delete", new { id=item.MunicipalityId }, new { @class = "btn btn-danger" })

</td>

</tr>

}

</table>

1. Add the Index post:

[HttpPost]

public ActionResult Index(MunicipalitiesView view)

{

var municipalities = db.Municipalities

.Include(m => m.Departament)

.OrderBy(m => m.Departament.Name)

.ThenBy(m => m.Name)

.ToList();

if (!string.IsNullOrEmpty(view.Department))

{

municipalities = municipalities.Where(m => m.Departament.Name.ToUpper().Contains(view.Department.ToUpper())).ToList();

}

if (!string.IsNullOrEmpty(view.Municipality))

{

municipalities = municipalities.Where(m => m.Name.ToUpper().Contains(view.Municipality.ToUpper())).ToList();

}

view.Municipalities = municipalities;

return View(view);

}

Para probar, mandar el origen de datos de un Cristal Report desde EF:

private ReportClass GenerateUserReport() { var Users = db.Users.ToList(); var report = new ReportClass(); report.FileName = Server.MapPath("~/Reports/CrystalReportUsers.rpt"); report.Load(); report.SetDataSource(Users); return report; }﻿

# MAIL HELPER

1. Add those keys to web.config y appSettings section:

<add key="AdminUser" value="taxesapplication@gmail.com" />

<add key="AdminPassWord" value="Taxes123." />

<add key="SMTPName" value="smtp.gmail.com" />

<add key="SMTPPort" value="587" />

1. Add the mail helper class:

public static async Task SendMail(string to, string subject, string body)

{

var message = new MailMessage();

message.To.Add(new MailAddress(to));

message.From = new MailAddress(WebConfigurationManager.AppSettings["AdminUser"]);

message.Subject = subject;

message.Body = body;

message.IsBodyHtml = true;

using (var smtp = new SmtpClient())

{

var credential = new NetworkCredential

{

UserName = WebConfigurationManager.AppSettings["AdminUser"],

Password = WebConfigurationManager.AppSettings["AdminPassWord"]

};

smtp.Credentials = credential;

smtp.Host = WebConfigurationManager.AppSettings["SMTPName"];

smtp.Port = int.Parse(WebConfigurationManager.AppSettings["SMTPPort"]);

smtp.EnableSsl = true;

await smtp.SendMailAsync(message);

}

}

public static async Task SendMail(List<string> mails, string subject, string body)

{

var message = new MailMessage();

foreach (var to in mails)

{

message.To.Add(new MailAddress(to));

}

message.From = new MailAddress(WebConfigurationManager.AppSettings["AdminUser"]);

message.Subject = subject;

message.Body = body;

message.IsBodyHtml = true;

using (var smtp = new SmtpClient())

{

var credential = new NetworkCredential

{

UserName = WebConfigurationManager.AppSettings["AdminUser"],

Password = WebConfigurationManager.AppSettings["AdminPassWord"]

};

smtp.Credentials = credential;

smtp.Host = WebConfigurationManager.AppSettings["SMTPName"];

smtp.Port = int.Parse(WebConfigurationManager.AppSettings["SMTPPort"]);

smtp.EnableSsl = true;

await smtp.SendMailAsync(message);

}

}

# USER HELPER

1. Add the user helper class:

public class UsersHelper : IDisposable

{

private static ApplicationDbContext userContext = new ApplicationDbContext();

private static ECommerceContext db = new ECommerceContext();

public static void CheckRole(string roleName)

{

var roleManager = new RoleManager<IdentityRole>(new RoleStore<IdentityRole>(userContext));

// Check to see if Role Exists, if not create it

if (!roleManager.RoleExists(roleName))

{

roleManager.Create(new IdentityRole(roleName));

}

}

public static void CheckSuperUser()

{

var userManager = new UserManager<ApplicationUser>(new UserStore<ApplicationUser>(userContext));

var email = WebConfigurationManager.AppSettings["AdminUser"];

var password = WebConfigurationManager.AppSettings["AdminPassWord"];

var userASP = userManager.FindByName(email);

if (userASP == null)

{

CreateUserASP(email, "Admin", password);

return;

}

userManager.AddToRole(userASP.Id, "Admin");

}

public static void CreateUserASP(string email, string roleName)

{

var userManager = new UserManager<ApplicationUser>(new UserStore<ApplicationUser>(userContext));

var userASP = new ApplicationUser

{

Email = email,

UserName = email,

};

userManager.Create(userASP, email);

userManager.AddToRole(userASP.Id, roleName);

}

public static void CreateUserASP(string email, string roleName, string password)

{

var userManager = new UserManager<ApplicationUser>(new UserStore<ApplicationUser>(userContext));

var userASP = new ApplicationUser

{

Email = email,

UserName = email,

};

userManager.Create(userASP, password);

userManager.AddToRole(userASP.Id, roleName);

}

public static async Task PasswordRecovery(string email)

{

var userManager = new UserManager<ApplicationUser>(new UserStore<ApplicationUser>(userContext));

var userASP = userManager.FindByEmail(email);

if (userASP == null)

{

return;

}

var user = db.TaxPaers.Where(tp => tp.UserName == email).FirstOrDefault();

if (user == null)

{

return;

}

var random = new Random();

var newPassword = string.Format("{0}{1}{2:04}\*",

user.FirstName.Trim().ToUpper().Substring(0, 1),

user.LastName.Trim().ToLower(),

random.Next(10000));

userManager.RemovePassword(userASP.Id);

userManager.AddPassword(userASP.Id, newPassword);

var subject = "Taxes Password Recovery";

var body = string.Format(@"

<h1>Taxes Password Recovery</h1>

<p>Yor new password is: <strong>{0}</strong></p>

<p>Please change it for one, that you remember easyly",

newPassword);

await MailHelper.SendMail(email, subject, body);

}

public void Dispose()

{

userContext.Dispose();

db.Dispose();

}

}

# USER MANAGEMENT

1. Add the user helper class:

public class UsersHelper : IDisposable

{

private static ApplicationDbContext userContext = new ApplicationDbContext();

private static ECommerceContext db = new ECommerceContext();

public static void CheckRole(string roleName)

{

var roleManager = new RoleManager<IdentityRole>(new RoleStore<IdentityRole>(userContext));

// Check to see if Role Exists, if not create it

if (!roleManager.RoleExists(roleName))

{

roleManager.Create(new IdentityRole(roleName));

}

}

public static void CheckSuperUser()

{

var userManager = new UserManager<ApplicationUser>(new UserStore<ApplicationUser>(userContext));

var email = WebConfigurationManager.AppSettings["AdminUser"];

var password = WebConfigurationManager.AppSettings["AdminPassWord"];

var userASP = userManager.FindByName(email);

if (userASP == null)

{

CreateUserASP(email, "Admin", password);

return;

}

userManager.AddToRole(userASP.Id, "Admin");

}

public static void CreateUserASP(string email, string roleName)

{

var userManager = new UserManager<ApplicationUser>(new UserStore<ApplicationUser>(userContext));

var userASP = new ApplicationUser

{

Email = email,

UserName = email,

};

userManager.Create(userASP, email);

userManager.AddToRole(userASP.Id, roleName);

}

public static void CreateUserASP(string email, string roleName, string password)

{

var userManager = new UserManager<ApplicationUser>(new UserStore<ApplicationUser>(userContext));

var userASP = new ApplicationUser

{

Email = email,

UserName = email,

};

userManager.Create(userASP, password);

userManager.AddToRole(userASP.Id, roleName);

}

public static async Task PasswordRecovery(string email)

{

var userManager = new UserManager<ApplicationUser>(new UserStore<ApplicationUser>(userContext));

var userASP = userManager.FindByEmail(email);

if (userASP == null)

{

return;

}

var user = db.Users.Where(tp => tp.UserName == email).FirstOrDefault();

if (user == null)

{

return;

}

var random = new Random();

var newPassword = string.Format("{0}{1}{2:04}\*",

user.FirstName.Trim().ToUpper().Substring(0, 1),

user.LastName.Trim().ToLower(),

random.Next(10000));

userManager.RemovePassword(userASP.Id);

userManager.AddPassword(userASP.Id, newPassword);

var subject = "Taxes Password Recovery";

var body = string.Format(@"

<h1>Taxes Password Recovery</h1>

<p>Yor new password is: <strong>{0}</strong></p>

<p>Please change it for one, that you remember easyly",

newPassword);

await MailHelper.SendMail(email, subject, body);

}

public void Dispose()

{

userContext.Dispose();

db.Dispose();

}

}

1. In the global.asax create the roles and super user:

private void CheckRolesAndSuperUser()

{

UsersHelper.CheckRole("Admin");

UsersHelper.CheckRole("User");

UsersHelper.CheckSuperUser();

}

1. When creates a new user, create the user as ASP User:

db.SaveChanges();

UsersHelper.CreateUserASP(user.UserName, "User");

# TO IDENTIFICATE MODEL ERRORS

try

{

db.SaveChanges();

}

catch (DbEntityValidationException e)

{

foreach (var eve in e.EntityValidationErrors)

{

Console.WriteLine("Entity of type \"{0}\" in state \"{1}\" has the following validation errors:",

eve.Entry.Entity.GetType().Name, eve.Entry.State);

foreach (var ve in eve.ValidationErrors)

{

Console.WriteLine("- Property: \"{0}\", Error: \"{1}\"",

ve.PropertyName, ve.ErrorMessage);

}

}

throw;

}

# IMPROVE FILE PICKER

1. Create in folder scripts the file: “fileupload.js”, with the following code:

$(document).ready(function () {

$("#files").on("change", handleFileSelect);

});

function handleFileSelect(e) {

var files = e.target.files;

var filesArr = Array.prototype.slice.call(files);

filesArr.forEach(function (f, item) {

if (f.type.match("image.\*")) {

var reader = new FileReader();

reader.readAsDataURL(f);

$("#NombreArchivo").empty();

$("#NombreArchivo").attr("title", f.name);

$("#NombreArchivo").append("<span class='glyphicon glyphicon-file kv-caption-icon' style='display:inline-block'></span>" + f.name);

}

else {

alert(f.name + ' no esta permitido cargarlo');

return;

}

});

}

1. Add this lines to “boostrap.css”:

.file-object {

margin: 0 0 -5px 0;

padding: 0;

}

.btn-file {

position: relative;

overflow: hidden;

}

.btn-file input[type=file] {

position: absolute;

top: 0;

right: 0;

min-width: 100%;

min-height: 100%;

text-align: right;

opacity: 0;

background: none repeat scroll 0 0 transparent;

cursor: inherit;

display: block;

}

.file-caption-name {

display: inline-block;

word-break: break-all;

display: inline-block;

height:auto;

}

.input-group-lg .file-caption-name {

height: auto;

}

.file-caption-disabled {

background-color: #EEEEEE;

cursor: not-allowed;

opacity: 1;

}

.file-actions, .file-other-error {

text-align: left;

}

.file-icon-lg {

font-size: 1.2em;

}

.file-icon-2x {

font-size: 2.4em;

}

.file-icon-4x {

font-size: 4.8em;

}

.file-input-new .file-preview, .file-input-new .close, .file-input-new .glyphicon-file,

.file-input-new .fileinput-remove-button, .file-input-new .fileinput-upload-button,

.file-input-ajax-new .fileinput-remove-button, .file-input-ajax-new .fileinput-upload-button {

display: none;

}

.file-actions {

margin-top: 15px;

}

.highlighted {

border: 2px dashed #999 !important;

background-color: #f0f0f0;

}

1. Register the new java script in “BundleConfig.cs”:

bundles.Add(new ScriptBundle("~/bundles/bootstrap").Include(

"~/Scripts/bootstrap.js",

"~/Scripts/respond.js",

"~/Scripts/fileupload.js"));

1. Change the file picker in the view:

<div class="form-group">

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

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

<div class="file-input-new">

<div class="input-group" style="width:280px">

<div tabindex="500" class="form-control file-caption">

<div class="file-caption-name" id="NombreArchivo">

</div>

</div>

<div class="input-group-btn" style="height:auto">

<div tabindex="500" class="btn btn-primary btn-file">

<i class="glyphicon glyphicon-folder-open"></i>

File Search...

@Html.TextBoxFor(modelo => modelo.LogoFile, new { type = "file", id = "files" })

</div>

</div>

</div>

</div>

</div>

</div>

# GENERIC CASCADE DROP DOWN LIST

1. Create in folder scripts the file: “ecommerce.js”, with the following code:

$(document).ready(function () {

$("#DepartmentId").change(function () {

$("#CityId").empty();

$("#CityId").append('<option value="0">[Select a city...]</option>');

$.ajax({

type: 'POST',

url: Url,

dataType: 'json',

data: { departmentId: $("#DepartmentId").val() },

success: function (data) {

$.each(data, function (i, data) {

$("#CityId").append('<option value="'

+ data.CityId + '">'

+ data.Name + '</option>');

});

},

error: function (ex) {

alert('Failed to retrieve cities.' + ex);

}

});

return false;

})

});

1. Register the new java script in “BundleConfig.cs”:

bundles.Add(new ScriptBundle("~/bundles/bootstrap").Include(

"~/Scripts/bootstrap.js",

"~/Scripts/respond.js",

"~/Scripts/fileupload.js",

"~/Scripts/ecommerce.js"));

1. In all views that uses cities, replace or put this lines in the scripts section:

@section Scripts {

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

<script type="text/javascript">

var Url = '@Url.Action("GetCities", "Generic")';

</script>

}

1. Now, we will create the generic controller, to avoid duplicated lines (and delete the method GetCities in other controllers):

public class GenericController : Controller

{

private ECommerceContext db = new ECommerceContext();

public JsonResult GetCities(int departmentId)

{

db.Configuration.ProxyCreationEnabled = false;

var cities = db.Cities.Where(c => c.DepartmentId == departmentId);

return Json(cities);

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

}

1. At least, to solve the problem to see all the first time, replace this method in the CombosHelper:

public static List<City> GetCities(int departmentId)

{

var cities = db.Cities.Where(c => c.DepartmentId == departmentId).ToList();

cities.Add(new City

{

CityId = 0,

Name = "[Select a city...]",

});

return cities.OrderBy(d => d.Name).ToList();

}

1. And replace all calls to GetCities that uses the new parameter. And test it!

# GENERIC VALIDATIONS FOR DUPLICATES AND DELETES

1. Create the class: “Response”, with the following code:

public class Response

{

public bool Succeeded { get; set; }

public string Message { get; set; }

}

1. Create the class: “DBHelper”, with the following code:

public static Response SaveChanges(ECommerceContext db)

{

try

{

db.SaveChanges();

return new Response { Succeeded = true, };

}

catch (Exception ex)

{

var response = new Response { Succeeded = false, };

if (ex.InnerException != null &&

ex.InnerException.InnerException != null &&

ex.InnerException.InnerException.Message.Contains("\_Index"))

{

response.Message = "There is a record with the same value";

}

else if (ex.InnerException != null &&

ex.InnerException.InnerException != null &&

ex.InnerException.InnerException.Message.Contains("REFERENCE"))

{

response.Message = "The record can't be delete because it has related records";

}

else

{

response.Message = ex.Message;

}

return response;

}

}

1. Replace all db.SaveChanges() by the following lines (example with categories):

public ActionResult Create(Category category)

{

if (ModelState.IsValid)

{

db.Categories.Add(category);

var response = DBHelper.SaveChanges(db);

if (response.Succeeded)

{

return RedirectToAction("Index");

}

ModelState.AddModelError(string.Empty, response.Message);

}

return View(category);

}

public ActionResult Edit(Category category)

{

if (ModelState.IsValid)

{

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

var response = DBHelper.SaveChanges(db);

if (response.Succeeded)

{

return RedirectToAction("Index");

}

ModelState.AddModelError(string.Empty, response.Message);

}

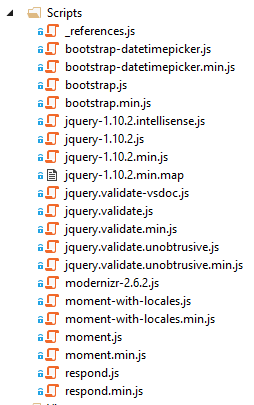
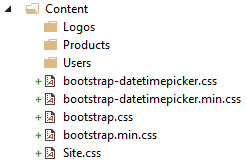
ViewBag.CompanyId = new SelectList(CombosHelper.GetCompanies(), "CompanyId", "Name", category.CompanyId);

return View(category);

}

# DATE PICKERS

1. Add the “datepicker” files, \*.js In Scripts folder and \*.css in the Content folder:

1. Register those files in the “BundleConfig.cs”:

bundles.Add(new ScriptBundle("~/bundles/bootstrap").Include(

"~/Scripts/moment.js",

"~/Scripts/bootstrap.js",

"~/Scripts/respond.js",

"~/Scripts/bootstrap-datetimepicker.js"));

bundles.Add(new StyleBundle("~/Content/css").Include(

"~/Content/bootstrap.css",

"~/Content/site.css",

"~/Content/bootstrap-datetimepicker.css"));

1. In the view that you use the date, replace this lines to the control:

<div class="form-group">

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

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

<div class='input-group date' id='datepicker' style="width:280px">

@Html.EditorFor(model => model.Date, new { htmlAttributes = new { @class = "form-control" } })

<span class="input-group-addon">

<span class="glyphicon glyphicon-calendar"></span>

</span>

</div>

@Html.ValidationMessageFor(model => model.Date, "", new { @class = "text-danger" })

</div>

</div>

1. And add this script:

<script type="text/javascript">

$(function () {

$('#datepicker').datetimepicker({ format: 'YYYY/MM/DD' });

});

</script>

1. To avoid problem in Google Chrome, you must delete the annotation: [DataType(DataType.Date)] in the model.

# MODAL DIALOGS

1. Change the line: @Html.ActionLink("Add Product", "AddProduct", new { }, new { @class = "btn btn-info" }), by:

@Html.ActionLink("Add Product", "AddProduct", "Orders", new { }, new { @class = "dialog-window btn btn-info" })

1. Add this lines before the scripts section:

<div class="modal fade" id="AddProduct" tabindex="-1" role="dialog" aria-labelledby="myModalLabel" aria-hidden="true" data-backdrop="false">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal" aria-hidden="true">&times;</button>

<h2 class="modal-title"></h2>

</div>

<div class="modal-body"><div class="te">Please wait...</div></div>

</div>

</div>

</div>

1. Add this lines in the script:

$(document).ready(function () {

$("body").on("click", "a.dialog-window", null, function (e) {

e.preventDefault();

var $link = $(this);

var title = $link.text();

$('#AddProduct .modal-title').html(title);

var url = $(this).attr('href');

if (url.indexOf('#') == 0) {

$('#AddProduct').modal('show');

}

else {

$.get(url, function (data) {

$('#AddProduct .te').html(data);

$('#AddProduct').modal();

}).success(function () { $('input:text:visible:first').focus(); });

}

});

});

1. Change the returns in GET and POST by partial view:

public ActionResult AddProduct()

{

var user = db.Users.Where(u => u.UserName == User.Identity.Name).FirstOrDefault();

ViewBag.ProductId = new SelectList(CombosHelper.GetProducts(user.CompanyId), "ProductId", "Description");

return PartialView();

}

1. Add this parameter to model:

[Display(Name = "Product", Prompt = "[Select a product...]")]

1. Change dropdown list by:

@Html.DropDownList("ProductId", null, ((IList<System.Web.Mvc.ModelMetadata>)ViewData.ModelMetadata.Properties)[0].Watermark, htmlAttributes: new { @class = "form-control", @required = "true" })

1. Add other GetProducts method y CombosHelper:

public static List<Product> GetProducts(int companyId, bool sw)

{

var products = db.Products.Where(p => p.CompanyId == companyId).ToList();

return products.OrderBy(p => p.Description).ToList();

}

1. In controller change the call to new method:

public ActionResult AddProduct()

{

var user = db.Users.Where(u => u.UserName == User.Identity.Name).FirstOrDefault();

ViewBag.ProductId = new SelectList(CombosHelper.GetProducts(user.CompanyId, false), "ProductId", "Description");

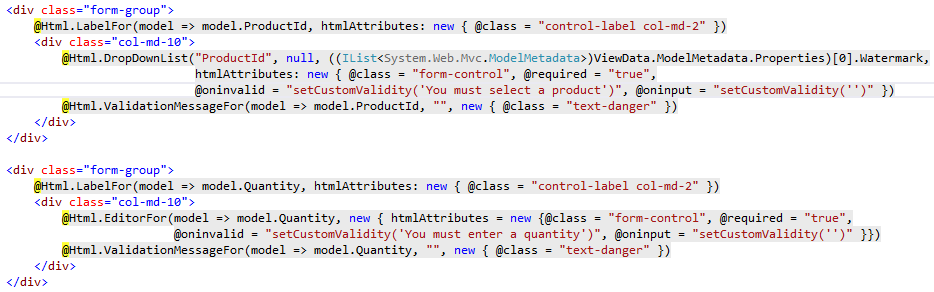
return PartialView();

}

1. Change in the view the quantity by:

@Html.EditorFor(model => model.Quantity, new { htmlAttributes = new { @class = "form-control", @required = "true" } })

1. To custom the messages you need to add the parameters: @oninvalid = "setCustomValidity('Your message…')", @oninput = "setCustomValidity('')". Example:



# JOIN SAMPLES

public class CompanyCustomer

{

[Key]

public int CompanyCustomerId { get; set; }

public int CompanyId { get; set; }

public int CustomerId { get; set; }

public virtual Company Company { get; set; }

public virtual Customer Customer { get; set; }

}

[HttpPost]

[ValidateAntiForgeryToken]

public ActionResult Create(Customer customer)

{

if (ModelState.IsValid)

{

using (var transaccion = db.Database.BeginTransaction())

{

try

{

db.Customers.Add(customer);

db.SaveChanges();

var user = db.Users.Where(u => u.UserName == User.Identity.Name).FirstOrDefault();

var companyCustomer = new CompanyCustomer

{

CompanyId = user.CompanyId,

CustomerId = customer.CustomerId,

};

db.CompanyCustomers.Add(companyCustomer);

db.SaveChanges();

UsersHelper.CreateUserASP(customer.UserName, "Customer");

transaccion.Commit();

return RedirectToAction("Index");

}

catch (Exception ex)

{

transaccion.Rollback();

ModelState.AddModelError(string.Empty, ex.Message);

}

}

}

ViewBag.CityId = new SelectList(CombosHelper.GetCities(customer.DepartmentId), "CityId", "Name", customer.CityId);

ViewBag.DepartmentId = new SelectList(CombosHelper.GetDepartments(), "DepartmentId", "Name", customer.DepartmentId);

return View(customer);

}

public ActionResult Index()

{

var user = db.Users.Where(u => u.UserName == User.Identity.Name).FirstOrDefault();

var qry = (from cu in db.Customers

join cc in db.CompanyCustomers on cu.CustomerId equals cc.CustomerId

join co in db.Companies on cc.CompanyId equals co.CompanyId

where co.CompanyId == user.CompanyId

select new { cu }).ToList();

var customers = new List<Customer>();

foreach (var item in qry)

{

customers.Add(item.cu);

}

return View(customers);

}

# LOGO IN MENU

1. Add this method in AccountController (don’t forget the context object and the IDispose implementation):

public void Logo(LoginViewModel model)

{

var user = db.Users.Where(u => u.UserName == model.Email).FirstOrDefault();

if (user != null)

{

var company = db.Companies.Find(user.CompanyId);

if (company != null)

{

Session["Logo"] = company.Logo;

}

}

}

1. Modify the Login Post method:

case SignInStatus.Success:

Logo(model);

return RedirectToLocal(returnUrl);

1. Modify the logoff method:

public ActionResult LogOff()

{

AuthenticationManager.SignOut(DefaultAuthenticationTypes.ApplicationCookie);

Session["Logo"] = null;

return RedirectToAction("Index", "Home");

}

1. Modify the: \_LoginPartial.cshtml by:

<ul class="nav navbar-nav navbar-right">

<li>

@if (Session["Logo"] != null)

{

<img src="@Url.Content(Session["Logo"].ToString())" alt="Image" style="width:50px;height:50px;max-width: 100%; height: auto; padding-top: 5px" />

}

</li>

<li>

@Html.ActionLink("Hello " + User.Identity.GetUserName() + "!", "ChangePassword", "Manage", routeValues: null, htmlAttributes: new { title = "Change Password" })

</li>

<li><a href="javascript:document.getElementById('logoutForm').submit()">Log off</a></li>

</ul>

# DELETE CONFIRM WITHOUT ALERT

1. Replace the link by this line:

<button type="button" class="dialog-window btn btn-danger" data-id="@item.ProductId">Delete</button>

1. Add the new div:

<div class="modal fade" id="DeleteProduct" tabindex="-1" role="dialog" aria-labelledby="myModalLabel" aria-hidden="true" data-backdrop="false">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<button type="button" class="close" data-dismiss="modal" aria-hidden="true">&times;</button>

<h2 class="modal-title">Delete Product</h2>

</div>

<div class="modal-body">

<div class="te">

Are you sure to delete the record?

</div>

<div id="response">

</div>

</div>

</div>

</div>

</div>

1. Add this lines in the script section:

$("body").on("click", "button.dialog-window", null, function (e) {

var htmlContent = '<br /><a href="@Url.Action("DeleteProduct", "Orders")/' + $(this).data('id') + '" class="btn btn-danger">Yes</a>&nbsp&nbsp&nbsp&nbsp<button type="button" id="btnNo" class="btn btn-success" data-dismiss="modal" aria-hidden="true">No</button>';

$("#response").html(htmlContent);

$('#DeleteProduct').modal('show');

});

Yo guardo los valores en el movil de esta forma una vez iniciada la sesión Application.Current.Properties["id"] = id.Text; Application.Current.Properties["pass"] = pass.Text; La proxima vez que el usuario desee entrar a la aplicación los obtengo de la siguiente forma string idProp = Application.Current.Properties.ContainsKey("id") ? Application.Current.Properties["id"] as string : ""; string passProp = Application.Current.Properties.ContainsKey("pass") ? Application.Current.Properties["pass"] as string : ""; Si no se encuentran vacios intento por debajo realizar la validación del login para verificar si esos datos aún son los del usuario, si veo que al consultar el servicio del login estos aún son los datos, dejo al usuario avanzar a las opciones de usuario como si estuviera logueado, si con los datos obtenidos del dispositivo no concuerdan al consumir el servicio, envío al usuario al login.﻿