Creación de aplicaciones MVC con EF

1. Crear un proyecto web asp.net entityframework, y seleccionar las opciones MVC, WebAPi y autenticación de usuarios individuales
2. Correr el proyecto para ver la funcionalidad que tiene (sin haber realizado nada)
3. Crear la cadena de conexión con la base de datos. Abrir el archivo webconfig y crear la cadena de conexión como se muestra a continuación. (Adsi\_2021)

<connectionStrings>

<add name="DefaultConnection" connectionString="Data Source=DESKTOP- M0E608B\SQLEXPRESS;Initial Catalog=Agenda2026693;Integrated Security=True"

providerName="System.Data.SqlClient" />

</connectionStrings>

1. Correr nuevamente el proyecto y revisar que la base de datos se cree en el servidor
2. Crear los modelos teniendo en cuenta que EntityFramework requiere una propiedad de tipo entera int, con nombreId, la cual será utilizada como clave primaria

public class Aprendiz

{

[Key]

public int AprendizId { get; set; } //identity

public string Documento { get; set; }

public string Nombres { get; set; }

public string Apellidos { get; set; }

public string Celular { get; set; }

public string Correo { get; set; }

}

1. Luego se debe crear la clase principal, que será la encargada de manejar las transacciones con la base de datos, esta clase se debe llamar Agenda2026693Context y heredar de la súper clase DbContext.

A esta clase se le crea un constructor por defecto que hace un llamado al constructor de la súper clase DbContext, enviándole como argumento el nombre de la conexión

public class Agenda2026693Context : DbContext

{

public Agenda2026693Context()

: base("DefaultConnection")

{

}

}

1. Luego a la clase Agenda2026693Context le debemos adicionar una propiedad de tipo DbSet, por cada una de las clases que se van adicionando al modelo, así

public class Agenda2026693Context : DbContext

{

public Agenda1Context()

: base("DefaultConnection")

{

}

public DbSet<Aprendiz> Aprendizs { get; set; }

}

1. Aquí se recomienda compilar el proyecto antes de empezar a crear los controladores
2. Para crear los controladores damos click derecho en la carpeta controller, new controlador y seleccionamos controlador vacío, el nombre del controlador es nombre del modelo pluralizado más la palaba controller

De forma original el controlador queda de la siguiente manera

public class AprendizsController : Controller

{

// GET: Aprendizs

public ActionResult Index()

{

return View();

}

}

1. El controlador es el encargado de manejar toda la lógica del negocio para ello se debe conectar con la base de datos, para poderse conectar se debe instanciar un objeto de tipo Agenda2026693Context

public class AprendicesController : Controller

{

private Agenda1Context db = new Agenda1Context();

// GET: Aprendices

public ActionResult Index()

{

return View();

}

}

1. También es muy recomendable cerrar la conexión con la base de datos al finalizar cada transacción, para ello adicionamos el siguiente método al controlador, el método dispose

public class AprendicesController : Controller

{

private Agenda1Context db = new Agenda1Context();

// GET: Aprendices

public ActionResult Index()

{

return View();

}

protected override void Dispose(bool disposing)

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

}

1. Al trabajar con MVC, debemos tener en cuenta el concepto de acciones, que son los diferentes eventos que se pueden presentar cuando el usuario interactúa con el proyecto y las diferentes respuestas o acciones que el proyecto realiza ante las dichas acciones.

La primera acción que se considera dentro del controlador, es la acción Index la cual se encarga de enviar a una vista un listado de todos los objetos que hay almacenado en base de datos (objetos de cada modelo), la lógica de la acción Index es la siguiente:

public ActionResult Index()

{

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

}

1. Ahora para poder que el usuario interactúe con la aplicación, debemos crear la vista para la acción determinada, para ello damos click derecho sobre el nombre de la acción y luego agregar vista.

Hay diferentes maneras de adicionar la vista; pero en este caso vamos a utilizar un aplantilla predefinida (créate, details, List, Delete) las plantillas predefinidas deben tener asociado un modelo y el contexto de la base de datos.

Las acciones de tipo Index por general se relacionan con la plantilla List, esta es la que vamos a seleccionar y obtendremos la siguiente vista en formato CSHTML

@model IEnumerable<Agenda1.Models.Aprendiz>

@{

ViewBag.Title = "Index";

}

<h2>Index</h2>

<p>

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

</p>

<table class="table">

<tr>

<th>

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

</th>

<th>

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

</th>

<th>

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

</th>

<th></th>

</tr>

@foreach (var item in Model) {

<tr>

<td>

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

</td>

<td>

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

</td>

<td>

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

</td>

<td>

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

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

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

</td>

</tr>

}

</table>

1. Para poder acceder a las nuevas acciones que se van adicionando al programa, deber ir actualizando el menú, para ello vamos a la carpeta vistas, subcarpeta shared, archivo \_layout.cshtml y adicional al menú las opciones que sean necesarias, de la siguiente manera

<div class="navbar-collapse collapse">

<ul class="nav navbar-nav">

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

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

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

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

</ul>

@Html.Partial("\_LoginPartial")

</div>

1. La siguiente acción es Create, la cual sirve para crear un registro; es importante tener en cuenta que esta acción tiene dos momentos, el primero se desencadena cuando el usuario da clic sobre el vínculo crear, ahí la aplicación lo debe enviar a una vista que le permita ingresar los datos del nuevo registro, este momento se denomina HttGet; luego cuando la nueva vista está llena, con los datos del nuevo registro se debe disparar una nueva acción en el momento HttPost, la cual captura los datos de la vista y los lleva a la base de datos por medio del controlador

[HttpGet]

public ActionResult Create()

{

return View();

}

1. Antes de generar la acción del HttPost, debemos crear la vista Create, ya sabemos que para ello damos clic derecho en el nombre de la acción, agregar vista, y para esta vista utilizaremos la plantilla créate, obviamente seleccionando el modelo y contexto adecuados

@model Agenda1.Models.Aprendiz

@{

ViewBag.Title = "Create";

}

<h2>Create</h2>

@using (Html.BeginForm())

{

@Html.AntiForgeryToken()

<div class="form-horizontal">

<h4>Aprendiz</h4>

<hr />

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

<div class="form-group">

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

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

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

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

</div>

</div>

<div class="form-group">

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

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

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

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

</div>

</div>

<div class="form-group">

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

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

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

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

</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")

</div>

@section Scripts {

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

}

1. El HttpPost de la acción Create es el siguiente

[HttpPost]

public ActionResult Create(Aprendiz aprendiz)

{

if (ModelState.IsValid)

{

db.Aprendices.Add(aprendiz);

db.SaveChanges();

}

else

{

return View(aprendiz);

}

return RedirectToAction("Index");

}