PREPARAR LA APLICACIÓN

Crear una nueva aplicación llamada *EntreLibros.*

Bajar de aulas las clases incluidas en el archivo 7zip en aulas e incluirlas en la carpeta Models. Se incluyen:

* Las clases del dominio: Libro, Tema, Comentario.
* La clase de contexto. Observar la sentencia que posibilita eliminar la pluralización de los nombres de las tablas (esto evitará que la tabla correspondiente a la entidad *Libro* se llame *Libroes*)*.*

protected override void OnModelCreating(DbModelBuilder modelBuilder)

{

modelBuilder.Conventions.Remove<PluralizingTableNameConvention>();

}

**NOTA**: Es necesario importar el espacio de nombres:

using System.Data.Entity.ModelConfiguration.Conventions;

Crear El Viewmodel

Crearemos un View Model para:

* Capturar todos los datos del libro.
* Exponer la lista de todos los temas disponibles en la biblioteca.
* Capturar el id del tema seleccionado como tema del libro.
* Capturar el archivo seleccionado como portada, y setearle el nombre del archivo en el libro, además de subirlo a la carpeta /Images/Fotos.

Crear una carpeta *ViewModels* debajo de la raíz de la aplicación.

Agregarle una clase LibroViewModel. El ViewModel servirá para las vistas que modifican el libro (create /edit) y permitirá:

using EntreLibros.Models;

namespace EntreLibros.ViewModels

{

public class LibroViewModel

{

public Libro UnLibro { get; set; }

public SelectList TodosLosTemas { get; set; }

public HttpPostedFileBase Archivo { get; set; }

public int IdTemaSeleccionado { get; set; }

public LibroViewModel()

{

this.UnLibro = new Libro();

cargarTemas();

}

}

}

Para poder utilizar la clase SelectList deberá importar el espacio de nombres:

using System.Web.Mvc;

Ahora prepararemos la SelectList, pasándole al constructor en primer lugar la lista de temas, como segundo parámetro la propiedad que será tomada como "value" de cada <option> y en tercer lugar la que será tomada como "text" de cada option.

private void cargarTemas()

{

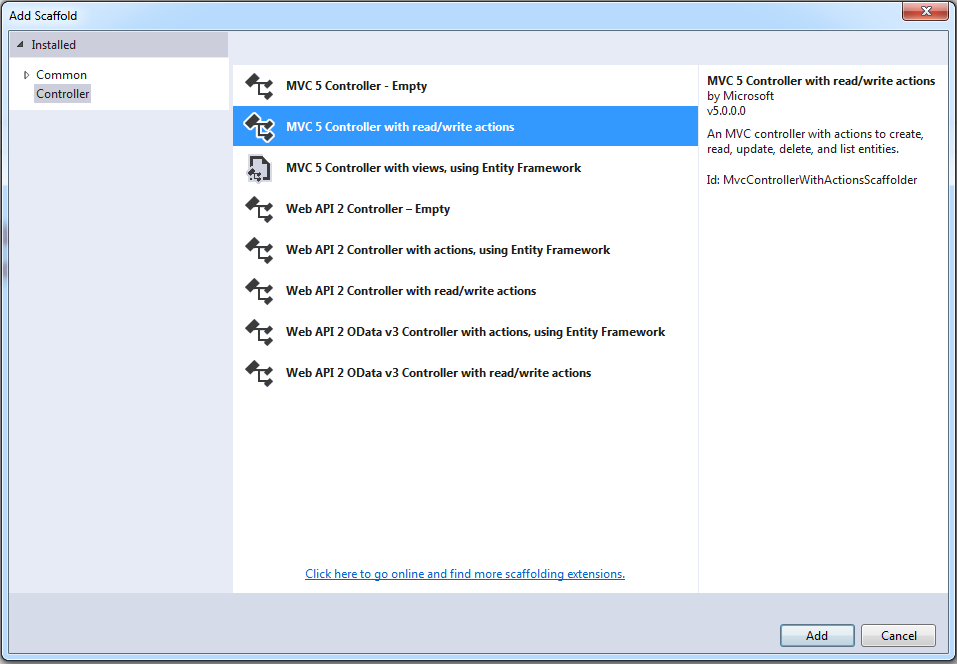
EntreLibrosContext db = new EntreLibrosContext();

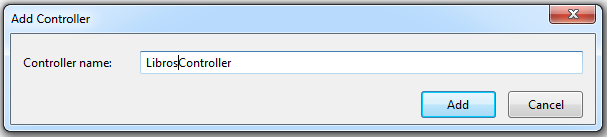
this.TodosLosTemas = new SelectList(db.Temas, "Id", "Nombre");

}

**IMPORTANTE: En este paso, antes de agregar el controlador y la vista no olvidar compilar el modelo (y el ViewModel) para que esté disponible para el scaffolding.**

Agregar El Controlador Para Los Libros

Agregar un controlador, LibrosController, eligiendo la plantilla con acciones de lectura y escritura, pero sin EntityFramework ni vista:  
  




Preparar el controlador la vista para crear un nuevo Libro:

En el método **Create** correspondiente al GET, retornaremos el ViewModel vacío, salvo la lista de temas que se desplegarán en una dropdownlist.

En el método correpondiente al POST, posiblitaremos que capture los datos del formulario en el View Model usando Model Binding, para posteriormente procesarlos y mapearlos en el modelo (Libro) para guardar en la base de datos, y que eventualmente los retorne a la vista en caso de que no se cumplan las reglas de validación:

//

// GET: /Libros/Create

[HttpGet]

public ActionResult Create()

{

return View(new EntreLibros.ViewModels.LibroViewModel());

}

//

// POST: /Libros/Create

[HttpPost]

public ActionResult Create(EntreLibros.ViewModels.LibroViewModel libroVM)

{

try

{

//

if (libroVM.Mapear())

{

Libro libro = libroVM.UnLibro;

if (libro != null)

{

EntreLibrosContext db = new EntreLibrosContext();

db.Libros.Add(libro);

db.SaveChanges();

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

}

}

return View(libroVM);

}

catch

{

return View();

}

}

AGREGAR EL CÓDIGO DE MAPEO:

En el View Model agregar el siguiente código para realizar el mapeo del libro:

public bool Mapear()

{

if (this.Archivo != null)

{

if (guardarArchivo(Archivo))

{

this.UnLibro.NombreArchivoPortada = Archivo.FileName;

EntreLibrosContext db = new EntreLibrosContext();

this.UnLibro.MiTema = db.Temas.Find(this.IdTemaSeleccionado);

return true;

}

}

return false;

}

Crear el método para procesar el archivo de imagen:

También en el View Model, agregar el método:

private bool guardarArchivo(HttpPostedFileBase archivo)

{

if (archivo != null)

{

string ruta = System.IO.Path.Combine

(AppDomain.CurrentDomain.BaseDirectory, "Images/fotos");

if (!System.IO.Directory.Exists(ruta))

System.IO.Directory.CreateDirectory(ruta);

ruta = System.IO.Path.Combine(ruta, archivo.FileName);

archivo.SaveAs(ruta);

return true;

}

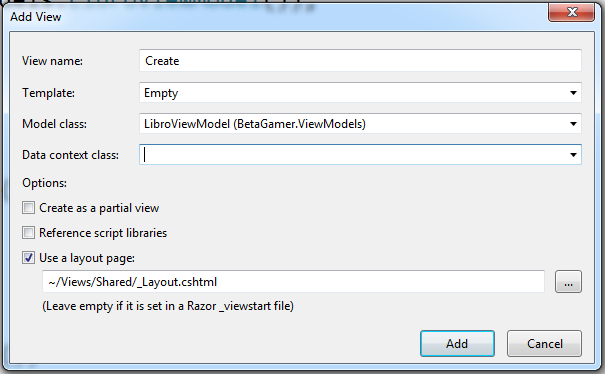
else return false;

}

Crear La Vista Para La Inserción De Un Nuevo Libro (A Partir Del Viewmodel)

Sobre la acción Create del controlador Libros, agregar una nueva vista (botón derecho).

Como plantilla para scaffold seleccionar Empty (Vacía), como modelo elegir el LibroViewModel, y usar la página maestra \_Layout.cshtml.



Se crea la vista sin el código para renderizar los elementos HTML:

@model EntreLibros.ViewModels.LibroViewModel

@{

ViewBag.Title = "Create";

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

}

<h2>Create</h2>

Como se observa, no incluye los Helpers, por lo que se deberán codificar a mano.

@model EntreLibros.ViewModels.LibroViewModel

@{

ViewBag.Title = "Create";

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

}

<h2>Agregar un nuevo libro al catálogo</h2>

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

@Html.AntiForgeryToken()

@Html.ValidationSummary(true)

Observar que se indicaron parámetros en el BeginForm (el método de acción, el controlador, el método HTTP Post, y el tipo de codificación aceptada.

<fieldset>

<legend>Libro</legend>

<div class="editor-label">

@Html.LabelFor(model => model.UnLibro.Titulo)

</div>

<div class="editor-field">

@Html.EditorFor(model => model.UnLibro.Titulo)

@Html.ValidationMessageFor(model => model.UnLibro.Titulo)

</div>

<div class="editor-label">

@Html.LabelFor(model => model.UnLibro.FechaPublicacion)

</div>

<div class="editor-field">

@Html.EditorFor(model => model.UnLibro.FechaPublicacion)

@Html.ValidationMessageFor(model => model.UnLibro.FechaPublicacion)

</div>

<div class="editor-label">

@Html.LabelFor(model => model. UnLibro.Descripción)

</div>

<div class="editor-field">

@Html.EditorFor(model => model. UnLibro.Descripción)

@Html.ValidationMessageFor(model => model.UnLibro.Descripción)

</div>

Se deben llamar igual el atributo del modelo y el name del input type file para que funcione el Model Binding.

<div class="editor-label">

@Html.LabelFor(model => model.Archivo)

</div>

<div class="editor-field">

<input type="file" name="Archivo" id="Archivo" />

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

</div>

<div>

<div class="editor-label">

@Html.LabelFor(model => model.IdTemaSeleccionado )

</div>

@Html.DropDownListFor(model => model.IdTemaSeleccionado , Model.TodosLosTemas )

</div>

<p>

<input type="submit" value="Create" />

</p>

</fieldset>

}

<div>

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

</div>

@section Scripts {

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

}

Código para el archivo de imagen

Observar que en el código anterior se incluyeron las líneas para el archivo:

<div class="editor-label">

@Html.LabelFor(model => model.Archivo)

</div>

<div class="editor-field">

Se deben llamar igual el atributo del modelo y el name del input

<input type="file" name="Archivo" id="Archivo" />

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

</div>

Código Para La Selección De Tema

También se incluyó el código que renderiza la dropdownlist de temas:

<div>

<div class="editor-label">

@Html.LabelFor(model => model.IdTemaSeleccionado )

</div>

@Html.DropDownListFor(model => model.IdTemaSeleccionado , Model.TodosLosTemas )

</div>

Para continuar:

PROBAR LA APLICACIÓN.

Modificar lo necesario para desplegar los libros con sus portadas.

**Sugerencia**: para desplegar una imagen se puede utilizar una sintaxis similar a este ejemplo usado para una entidad Persona con atributos Nombre, FechaNacimiento y Foto:

@foreach (var item in Model) {

<tr>

<td>

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

</td>

<td>

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

</td>

<td>

<img src="@Url.Content("~/Images/Fotos/"+item.ArchivoFoto )" width="40%" height="auto" alt="foto"/>

</td>

<td>

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

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

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

</td>

</tr>

}