preparar la aplicación

Utilizaremos como base la aplicación *EntreLibros* creada en el laboratorio 06*.*

Dentro de la carpeta Models agregaremos una nueva clase Autor(Id, Nombre) y agregaremos en la clase Libro una propiedad List<Autor> de nombre Autores.

Actualizaremos la clase de contexto agregando un DbSet para el manejo de esta nueva clase.

actualizar los datos

Abriremos la consola de administración de paquetes y agregaremos la migración. Si no lo ha hecho ya, primero deberá habilitar las migraciones con el comando: Enable-Migrations.

Luego ejecute los siguientes comandos:

Add-Migration claseAutor  
Update-Database

Desde el manager de SQLServer, verifique que los cambios de la migración se impactaron en la BD y agregue algunos autores en la tabla respectiva para poder hacer pruebas.

actualizar el viewmodel

En el viewmodel que habíamos creado para la vista de creación de libros agregamos dos atributos nuevos (los que están en negrita):

using EntreLibros.Models;

namespace EntreLibros.ViewModels

{

public class LibroViewModel

{

public Libro UnLibro { get; set; }

public SelectList TodosLosTemas { get; set; }

**public MultiSelectList TodosLosAutores { get; set; }**

public HttpPostedFileBase Archivo { get; set; }

public int IdTemaSeleccionado { get; set; }

**public int[] IdsAutoresSeleccionados { get; set; }**

public LibroViewModel()

{

this.UnLibro = new Libro();

cargarTemas();

**cargarAutores();**

}

}

}

La clase MultiSelectList al igual que la clase SelectList se encuentra dentro del espacio de nombres System.Web.Mvc

Esta clase nos permitirá generar una ListBox de selección múltiple con todos los autores. Los ids de los autores seleccionados se guardarán en el vector IdsAutoresSeleccionados para su posterior recuperación utilizando el contexto.

También agregamos en el constructor la invocación a un método de carga de autores similar al que hicimos para la carga de temas. El método es el siguiente:

private void cargarAutores()

{

EntreLibrosContext db = new EntreLibrosContext();

this.TodosLosAutores = new MultiSelectList(db.Autores, "Id", "Nombre");

}

Al igual que con la clase SelectList el constructor de la clase MultiSelectList acepta como primer parámetro la colección, como segundo parámetro la propiedad que será tomada como "value" de cada <option> (el DataValueField del ListBox) y en tercer lugar la que será tomada como "text" de cada option (el DataTextField del ListBox).

Por último, agregaremos el código necesario para mapear los autores (en engrita):

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);

**this.UnLibro.Autores = new List<Autor>();**

**foreach (int idAutor in this.IdsAutoresSeleccionados)**

**{**

**this.UnLibro.Autores.Add(db.Autors.Find(idAutor));**

**}**

return true;

}

}

return false;

}

**En este paso, antes de actualizar la vista compilar el ViewModel para que esté disponible para el scaffolding.**

actualizar el controlador:

En el método correpondiente al POST, agregaremos el siguiente código que evita que se creen duplicados de los autores y temas ya existentes (en negrita) :

// 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();

**//Para evitar que se creen nuevamente temas**

**//y autores ya existentes:**

**db.Entry(libro.MiTema).State = EntityState.Unchanged;**

**foreach(Autor autor in libro.Autores)**

**{**

**db.Entry(autor).State = EntityState.Unchanged;**

**}**

db.Libros.Add(l);

db.SaveChanges();

return RedirectToAction("Index");

}

}

return View(libroVM);

}

catch(Exception ex)

{

return View();

}

}

actualizar la vista:

Incluiremos el código que renderiza la ListBox de autores (en algún lugar al final, pero dentro del formulario):

<div>

<div class="editor-label">

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

</div>

@Html.ListBoxFor(model => model.IdsAutoresSeleccionados , Model.TodosLosAutores )

</div>

Para continuar:

PROBAR LA APLICACIÓN.

Nos va a dar un error al ejecutarse la acción. Al debuguear el código de la acción HttpPost Create del controlador LibrosController encontramos que el error es el siguiente:   
*"Varias instancias de IEntityChangeTracker no pueden hacer referencia a un objeto entidad."*

solucionar el problema:

Este problema ocurre cuando existen varios contextos referenciando una misma entidad. Una buena práctica para evitar estas situaciones es siempre colocar el contexto dentro de un bloque using para asegurarnos de que se haga el dispose del objeto al terminar de usarlo. Cambiamos entonces el código del ViewModel y del Controllador para usar bloques using:

public bool Mapear()

{

if (this.Archivo != null)

{

if (guardarArchivo(Archivo))

{

this.UnLibro.NombreArchivoPortada = Archivo.FileName;

**using (EntreLibrosContext db = new EntreLibrosContext())  
 {**

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

this.UnLibro.Autores = new List<Autor>();

foreach (int idAutor in this.IdsAutoresSeleccionados)

{

this.UnLibro.Autores.Add(db.Autors.Find(idAutor));

}

**}**

return true;

}

}

return false;

}

[HttpPost]

public ActionResult Create(EntreLibros.ViewModels.LibroViewModel libroVM)

{

try

{

if (libroVM.Mapear())

{

Libro libro = libroVM.UnLibro;

if (libro != null)

{

**using (EntreLibrosContext db = new EntreLibrosContext())  
 {**

**//Para evitar que se creen nuevamente temas**

**//y autores ya existentes:**

db.Entry(libro.MiTema).State = EntityState.Unchanged;

foreach(Autor autor in libro.Autores)

{

db.Entry(autor).State = EntityState.Unchanged;

}

db.Libros.Add(l);

db.SaveChanges();  
 }

return RedirectToAction("Index");

**}**

}

return View(libroVM);

}

catch(Exception ex)

{

return View();

}

}

PROBAMOS NUEVAMENTE LA APLICACIÓN PARA CORROBORAR QUE EL ERROR SE SOLUCIONÓ Y QUE LOS DATOS DEL NUEVO LIBRO QUEDARON CORRECTAMENTE GUARDADOS EN LAS TABLAS CORRESPONDIENTES DE LA BD.