preparar la aplicación

Utilizaremos como base la aplicación *EntreLibros* creada en el laboratorio 06 y modificada en el laboratorio 08*.*

actualizar el controlador:

Modificaremos la acción Index del controlador LibrosController para que le pase a la vista como modelo una List<Libro> con todos los libros existentes:

public ActionResult Index()

{

EntreLibrosContext db = new EntreLibrosContext();

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

}

actualizar la vista:

A la vista Index.cshtml le agregaremos lo necesario para que reciba una colección de Libros como modelo y despliegue la información de los libros en una tabla generada dinámicamente usando Razor:

@model IEnumerable<EntreLibros.Models.Libro>

@{

ViewBag.Title = "Index";

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

}

<h2>Index</h2>

<p>

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

</p>

<table class="table">

<tr>

<th>

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

</th>

<th>

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

</th>

<th>

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

</th>

<th></th>

</tr>

@foreach (var item in Model)

{

<tr>

<td>

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

</td>

<td>

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

</td>

<td>

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

</td>

<td>

@Html.ActionLink("Ver Detalle en página detalle", "Details",

new { id = item.Id })

</td>

</tr>

}

</table>

Actualizar la acción para ver detalle de libros:

Agregar el siguiente código a la acción Detail del controlador LibrosController:

public ActionResult Details(int? id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Libro libro = db.Libros.Find(id);

if (libro == null)

{

return HttpNotFound();

}

return View(libro);

}

Actualizar la vista de detalle (details.cshtml):

Actualizamos la vista para que reciba como modelo un Libro y muestre sus datos:

@model EntreLibros.Models.Libro

@{

ViewBag.Title = "Details";

}

<h2>Detalle</h2>

<div>

<h4>Libro</h4>

<hr />

<dl class="dl-horizontal">

<dt>

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

</dt>

<dd>

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

</dd>

<dt>

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

</dt>

<dd>

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

</dd>

<dt>

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

</dt>

<dd>

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

</dd>

</dl>

</div>

<p>

@Html.ActionLink("Volver al listado", "Index")

</p>

PROBAR LA APLICACIÓN.

Usando el helper para grillas (webgrid):

Ahora mostraremos todos los libros en la vista Index, pero en vez de generar la tabla manualmente con Razor, lo haremos con la ayuda del html helper para grillas: WebGrid.

Agregaremos el siguiente código en la vista, inmediatamente debajo de la tabla que generamos con Razor:

<div id="gridContent" class="table">

<h2>La Grilla:</h2>

@{

var grid = new WebGrid(

Model,

canPage: true,

rowsPerPage: 5,

selectionFieldName: "seleccionado"

);

grid.Pager(WebGridPagerModes.NextPrevious);

}

@grid.GetHtml(

columns: grid.Columns(

grid.Column(header: "Título", columnName: "Texto"),

grid.Column(header: "Publicado el", columnName: "FechaPublicacion",

format: (item) => item.Fecha.ToShortDateString()),

grid.Column(header: "Descripción", columnName: " Descripción "),

grid.Column(header: "Ver detalle en página detalle",

format: (item) => Html.ActionLink("Ver", "Details", new { id = item.Id }))

)

)

</div>

**Notas:**

Lo primero que hace es crear un objeto de la clase WebGrid al cual se le configura su datasource (Model), si admite paginado, la cantidad de filas por página, etc. El constructor admite varios parámetros opcionales que permiten configurar varios aspectos de la grilla. Por más información puede consultar la documentación en la MSDN:

<https://msdn.microsoft.com/en-us/library/system.web.helpers.webgrid.webgrid(v=vs.111).aspx>

Luego configura el tipo de paginado mediante el método Pager que recibe un enumerado.

El método GetHtml genera el código html de una tabla y es posible determinar qué columnas mostrar y la manera en que se muestra la información, mediante los parámetros que se le pasan. Este método admite varios parámetros opcionales. El link a la documentación de la MSDN es el siguiente:

<https://msdn.microsoft.com/en-us/library/system.web.helpers.webgrid.gethtml(v=vs.111).aspx>

En nuestro caso le decimos qué columnas mostrar y a qué campos del origen de datos se mapean.

Observe que es posible formatear la información (columna fecha) e incluso crear dinámicamente links a partir de los datos del ítem vinculado a cada fila (columna para ver detalle).

PRUEBE LA APLICACIÓN.

Obtener el objeto de la selección para mostrar detalle:

Es posible obtener el ítem vinculado a la fila seleccionada de un WebGrid. Eso nos permitirá hacer cualquier cosa que querramos con él. En este caso, lo que haremos será mostrar su información en la misma página, debajo de la grilla.

Agregaremos en el método GetHtml del WebGrid una columna más para realizar esta acción de selección. El código es el siguiente (negrita):

@grid.GetHtml(

columns: grid.Columns(

**grid.Column("Ver detalle abajo", format: (item) => item.GetSelectLink("Ver")),**

grid.Column(header: "Título", columnName: "Titulo"),

grid.Column(header: "Publicado el", columnName: "FechaPublicacion",

format: (item) => item.FechaPublicacion.ToShortDateString()),

grid.Column(header: "Descripción", columnName: "Descripción"),

grid.Column(header: "Ver detalle en página detalle",

format: (item) => Html.ActionLink("Ver", "Details", new { id = item.Id }))

)

)

Debajo de la div donde generamos la grilla agregaremos una nueva div con el siguiente código:

<div>

@{

EntreLibros.Models.Libro libro = new EntreLibros.Models.Libro();

}

@if (grid.HasSelection)

{

<h2>Seleccionado:</h2>

libro = (EntreLibros.Models.Libro)grid.Rows[grid.SelectedIndex].Value;

<b>Id </b> @libro.Id<br />

<b>Título </b> @libro.Titulo<br />

<b>Descripción </b> @libro.Descripción<br />

<b>Fecha publicado </b> @libro.FechaPublicacion.ToShortDateString()<br />

}

</div>

Note que la información se mostrará solamente cuando existe algún ítem seleccionado en la grilla. Por ese motivo no aparece detalle cuando la vista se carga por primera vez.

PROBAR LA APLICACIÓN