Aquí tienes un ejemplo de cómo podrías crear las vistas de creación y edición para la clase `Impresora` utilizando DropDownList en cascada para las tablas `Marca`, `Modelo` e `Impresora` en ASP.NET Core MVC con Entity Framework Core:

```csharp

// Controlador

public class ImpresorasController : Controller

{

private readonly ApplicationDbContext \_context;

public ImpresorasController(ApplicationDbContext context)

{

\_context = context;

}

// GET: Impresoras/Create

public IActionResult Create()

{

ViewData["EstadoId"] = new SelectList(\_context.Estados, "Id", "Nombre");

ViewData["MarcaId"] = new SelectList(\_context.Marcas, "Id", "Nombre");

ViewData["ModeloId"] = new SelectList(\_context.Modelos.Where(m => m.MarcaId == 0), "Id", "Nombre");

return View();

}

// POST: Impresoras/Create

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create([Bind("Id,CodigoActivoFijo,MarcaId,ModeloId,EstadoId,DireccionIP")] Impresora impresora)

{

if (ModelState.IsValid)

{

\_context.Add(impresora);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

ViewData["EstadoId"] = new SelectList(\_context.Estados, "Id", "Nombre", impresora.EstadoId);

ViewData["MarcaId"] = new SelectList(\_context.Marcas, "Id", "Nombre", impresora.MarcaId);

ViewData["ModeloId"] = new SelectList(\_context.Modelos.Where(m => m.MarcaId == impresora.MarcaId), "Id", "Nombre", impresora.ModeloId);

return View(impresora);

}

// GET: Impresoras/Edit/5

public async Task<IActionResult> Edit(int? id)

{

if (id == null)

{

return NotFound();

}

var impresora = await \_context.Impresoras.FindAsync(id);

if (impresora == null)

{

return NotFound();

}

ViewData["EstadoId"] = new SelectList(\_context.Estados, "Id", "Nombre", impresora.EstadoId);

ViewData["MarcaId"] = new SelectList(\_context.Marcas, "Id", "Nombre", impresora.MarcaId);

ViewData["ModeloId"] = new SelectList(\_context.Modelos.Where(m => m.MarcaId == impresora.MarcaId), "Id", "Nombre", impresora.ModeloId);

return View(impresora);

}

// POST: Impresoras/Edit/5

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, [Bind("Id,CodigoActivoFijo,MarcaId,ModeloId,EstadoId,DireccionIP")] Impresora impresora)

{

if (id != impresora.Id)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(impresora);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!ImpresoraExists(impresora.Id))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

ViewData["EstadoId"] = new SelectList(\_context.Estados, "Id", "Nombre", impresora.EstadoId);

ViewData["MarcaId"] = new SelectList(\_context.Marcas, "Id", "Nombre", impresora.MarcaId);

ViewData["ModeloId"] = new SelectList(\_context.Modelos.Where(m => m.MarcaId == impresora.MarcaI));

Aquí tienes un ejemplo de cómo podrías crear las vistas de creación y edición para la clase `Impresora` en ASP.NET Core MVC utilizando Razor y el código del controlador que te proporcioné anteriormente:

```html

@model GestionImpresoras.Models.Impresora

@{

ViewData["Title"] = "Crear Impresora";

}

<h1>@ViewData["Title"]</h1>

<h4>Impresora</h4>

<hr />

<div class="row">

<div class="col-md-4">

<form asp-action="Create">

<div asp-validation-summary="ModelOnly" class="text-danger"></div>

<div class="form-group">

<label asp-for="CodigoActivoFijo" class="control-label"></label>

<input asp-for="CodigoActivoFijo" class="form-control" />

<span asp-validation-for="CodigoActivoFijo" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="MarcaId" class="control-label"></label>

<select asp-for="MarcaId" class ="form-control" asp-items="ViewBag.MarcaId"></select>

<span asp-validation-for="MarcaId" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="ModeloId" class="control-label"></label>

<select asp-for="ModeloId" class ="form-control" asp-items="ViewBag.ModeloId"></select>

<span asp-validation-for="ModeloId" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="EstadoId" class="control-label"></label>

<select asp-for="EstadoId" class ="form-control" asp-items="ViewBag.EstadoId"></select>

<span asp-validation-for="EstadoId" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="DireccionIP" class="control-label"></label>

<input asp-for="DireccionIP" class="form-control" />

<span asp-validation-for="DireccionIP" class="text-danger"></span>

</div>

<div class="form-group">

<input type="submit" value="@ViewData["Title"]" class="btn btn-primary" />

</div>

</form>

</div>

</div>

<div>

<a asp-action="Index">Volver a la lista</a>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

```

Este es un ejemplo de cómo podrías crear la vista de creación para la clase `Impresora`. La vista de edición sería muy similar, solo tendrías que cambiar el `asp-action` del formulario a `"Edit"` y actualizar el título y el valor del botón de envío.

También puedes agregar validación del lado del cliente utilizando jQuery y la biblioteca de validación de jQuery. Para hacerlo, asegúrate de incluir las referencias necesarias a estas bibliotecas en tu vista o en tu archivo `\_Layout.cshtml` y luego incluye el siguiente código en tu vista:

```javascript

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

<script type= "text/javascript">

$(document).ready(function () {

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

$.get("/Impresoras/GetModelos", { MarcaId: $("#MarcaId").val() }, function (data) {

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

$.each(data, function (index, row) {

$("#ModeloId").append("<option value='" + row.id + "'>" + row.nombre + "</option>")

});

});

})

});

</script>

}

```

Este código agrega un controlador de eventos al elemento `select` con `id` `"MarcaId"` para que cuando cambie su valor, se realice una solicitud GET al método `GetModelos` del controlador `Impresoras` pasando el valor seleccionado como parámetro. Luego, se actualiza el elemento `select` con `id` `"ModeloId"` con los datos recibidos.

Recuerda que también debes agregar el método `GetModelos` en tu controlador `Impresoras` para que este código funcione correctamente. Este método debería devolver una lista de modelos filtrados por la marca seleccionada en formato JSON.

```csharp

public JsonResult GetModelos(int MarcaId)

{

var modelos = \_context.Modelos.Where(m => m.MarcaId == MarcaId).Select(m => new { id = m.Id, nombre = m.Nombre }).ToList();

return Json(modelos);

}

```

Sí, puedes utilizar `ViewBag` para pasar datos desde el controlador a la vista. En el ejemplo de código que te proporcioné anteriormente, ya se está utilizando `ViewBag` para pasar las listas de marcas, modelos y estados desde el controlador a la vista.

Por ejemplo, en el método `Create` del controlador `Impresoras`, se están creando tres listas de selección para las propiedades `MarcaId`, `ModeloId` y `EstadoId` de la clase `Impresora` y se están almacenando en `ViewBag`:

```csharp

ViewData["EstadoId"] = new SelectList(\_context.Estados, "Id", "Nombre");

ViewData["MarcaId"] = new SelectList(\_context.Marcas, "Id", "Nombre");

ViewData["ModeloId"] = new SelectList(\_context.Modelos.Where(m => m.MarcaId == 0), "Id", "Nombre");

```

Luego, en la vista de creación, se están utilizando estos datos almacenados en `ViewBag` para generar los elementos `select` para estas propiedades utilizando el método de extensión `asp-items`:

```html

<div class="form-group">

<label asp-for="MarcaId" class="control-label"></label>

<select asp-for="MarcaId" class ="form-control" asp-items="ViewBag.MarcaId"></select>

<span asp-validation-for="MarcaId" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="ModeloId" class="control-label"></label>

<select asp-for="ModeloId" class ="form-control" asp-items="ViewBag.ModeloId"></select>

<span asp-validation-for="ModeloId" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="EstadoId" class="control-label"></label>

<select asp-for="EstadoId" class ="form-control" asp-items="ViewBag.EstadoId"></select>

<span asp-validation-for="EstadoId" class="text-danger"></span>

</div>

```

De esta manera, puedes utilizar `ViewBag` para pasar datos desde el controlador a la vista y utilizarlos para generar contenido dinámico en la vista.