L6

UNIVERSIDAD CENFOTEC_ / CURSO: BISOFT-13

PROFESOR: DENNIS CÓRDOBA LÓPEZ

LABORATORIO 6

El desarrollo del siguiente laboratorio debe hacerse de manera individual.

PARTE 1:

Desarrolle los siguientes controles:

- CtrlInput
- CtrlTable
- CtrlCombobox
- CtrlButton

CONTROLES

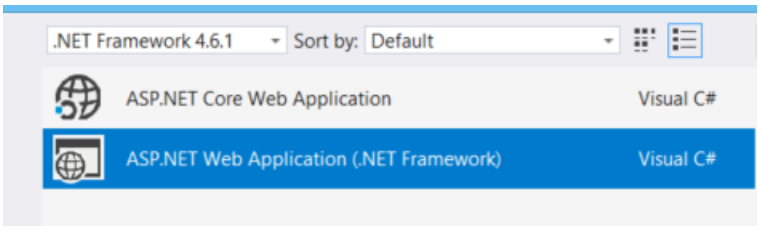
CONTROL HTML: Definimos un control a un elemento o componente visual de html que tiene un comportamiento predefinido, al ser un componente este puede ser reutilizable debido a que el mismo es parametrizable.

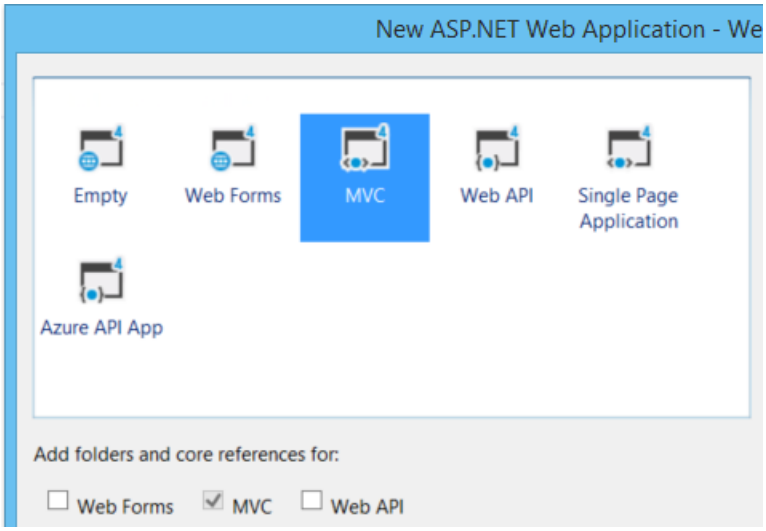
Los elementos html en sí mismos son controles, en nuestro caso vamos a tomar esos elementos html y "extenderlos" para crear nuestros propios controles alineados a nuestra arquitectura y al comportamiento de la misma.

ELABORACION DE CONTROLES

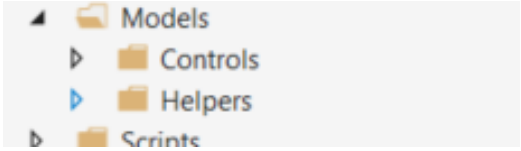
Para la elaboración de controles debe ejecutar una serie de pasos, los mismos ya han sido ejecutados en el laboratorio de ejemplo, donde ya se creó el control tabla.

1. Creación de un proyecto web, de tipo MVC.





2. Una vez creado el proyecto, encontrará una distribución de carpetas, donde destacan las carpetas para las Vistas, Controladores y Modelos, esto porque es el modelo que estamos utilizando MVC.
3. Encontrará dos carpetas dentro de Models, una llamada Helpers y otra llamada Controls.



4. Dentro de la carpeta Helpers, encontraremos una clase llamada ControlExtensions.cs, en esta clase se encuentra la definición de los controles y su manera de implementación.
   Cada control que se define cuenta con un método estatico que retorna el texto del html correspondiente a dicho control, este texto es variable según la parametrización.

5. Metodo CtrlTable:

   Este método será invocado cuando se desee instanciar el control tabla, este método recibe una serie de parámetros que definen como se visualiza y su comportamiento.

```
public static class ControlExtensions
{
    public static HtmlString CtrlTable(this HtmlHelper html, string id, string title,
        string columnsTitle, string ColumnsDataName)
    {
        var ctrl = new CtrlTableModel
        {
            Id = id,
            Title = title,
            Columns = columnsTitle,
            ColumnsDataName = ColumnsDataName
        };

        return new HtmlString(ctrl.GetHtml());
    }

}
```

6. CtrlTableModel, esta clase está siendo usada en la imagen anterior, la misma se ubica en la carpeta Controls, esta clase representa el control tabla del lado C# de la arquitectura, el modelo es el encargado de producir el HTML correspondiente.

```
public class CtrlTableModel : CtrlBaseModel
{
    public CtrlTableModel()
    {
        Columns = "";
    }

    public string Title { get; set; }
    public string Columns { get; set; }
    public string ColumnsDataName { get; set; }

    public int ColumnsCount => Columns.Split(',').Length;

    public string ColumnHeaders
    {
        get
        {
            var headers = "";
            foreach(var text in Columns.Split(','))
            {
                headers += "<th>" + text + "</th>";
            }

            return headers;
        }
    }

}
```

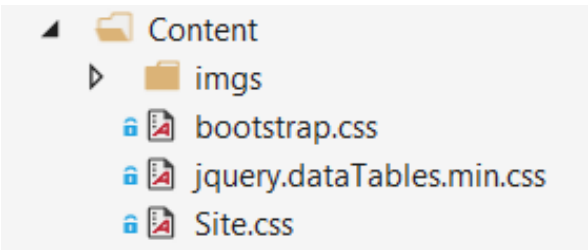
7. La clase CtrlTableModel, extiende de CtrlBaseModel, CtrlBaseModel es la clase “padre” de todo control que implementemos, dicha clase cuenta con un métodos y atributos generales de todo control, entre ellos el método GetHtml.

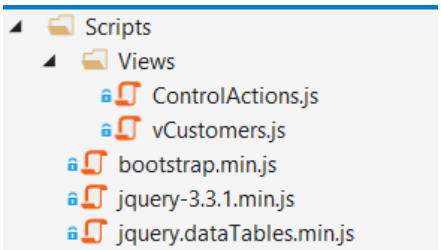
```
public class CtrlBaseModel
{

    public string Id { get; set; }


    private string ReadFileText()...

    public string GetHtml()
    {
        var html = ReadFileText();

        foreach (var prop in this.GetType().GetProperties())
        {
            var value = prop.GetValue(this, null).ToString();

            var tag = string.Format("-#{0}-", prop.Name);
            html = html.Replace(tag, value);
        }
        return html;
    }
}
```

8. El método GetHtml es el encargado generar el html que será enviado a la vista, esto lo hace parseando el archivo CtrlTableModel.html, este parseo consiste en inyectar al html del control los atributos correspondientes. Debe existir un archivo html por cada uno de los controles.

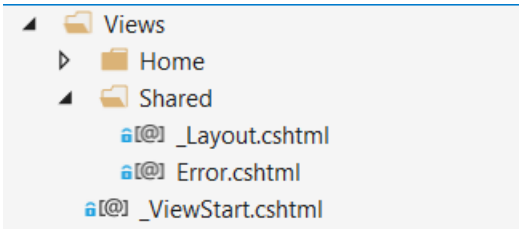
```
<script>
    $(document).ready(function () {
        $('#-#Id-').DataTable();
    });
</script>
<div class="card border-secondary mb-6">
    <div class="card-header">-#Title-</div>
    <div class="card-body">
        <table id="-#Id-" class="display" cellspacing="0" width="100%"
               ColumnsDataName="-#ColumnsDataName-">
            <thead>
                <tr>
                    -#ColumnHeaders-
                </tr>
            </thead>
            <tbody>
            </tbody>
        </table>
    </div>
</div>
```

9. En el caso del control tabla se requiere importar un conjunto dependencias, css y js, estas librerias deben agregarse al proyecto en la carpeta Content y Scripts según corresponda.





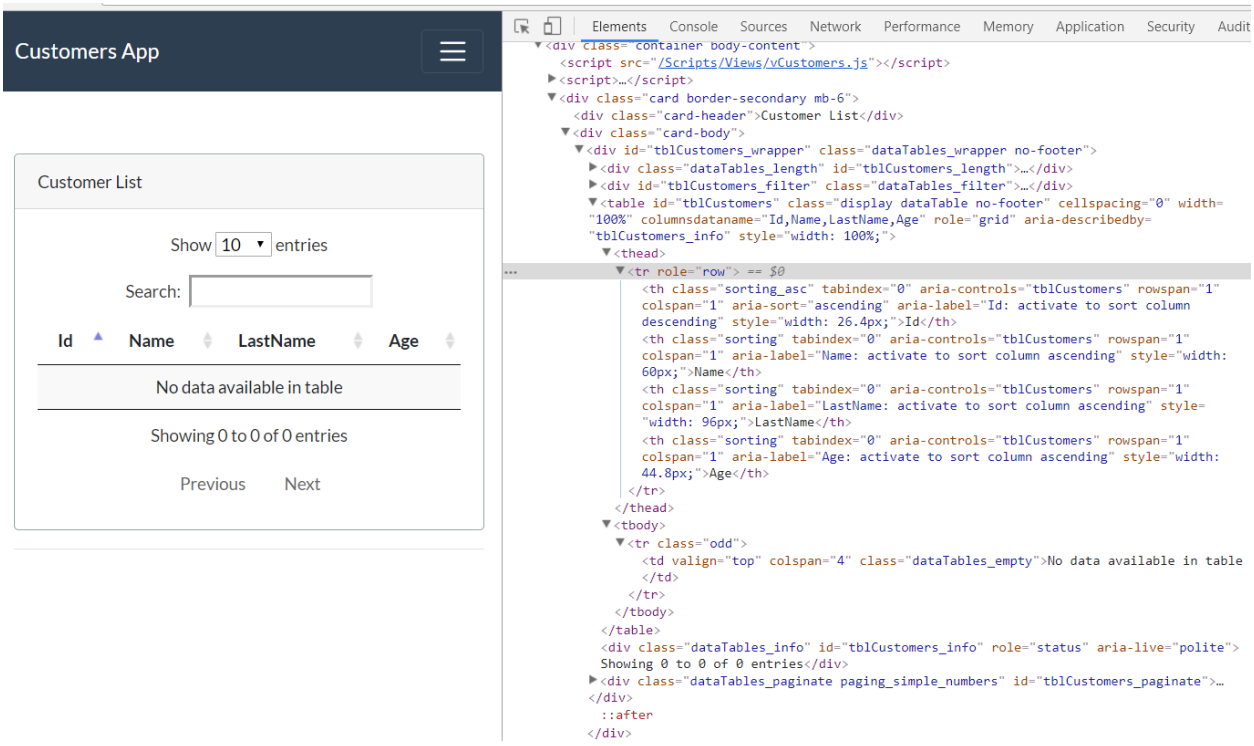
10. Una vez que las dependencias son agregadas, se deben importar en el proyecto en el archivo _layout, que esta presente en todas las vistas.



11. Una vez construido el control tabla, se puede invocar el mismo desde diversas vistas de manera paramétrica.

```
@using WebApp.Helpers;
<script src="~/Scripts/Views/vCustomers.js"></script>
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
}


@Html.CtrlTable("tblCustomers", title: "Customer List",
    columnsTitle: "Id,Name,LastName,Age", ColumnsDataName: "Id,Name,LastName,Age")
```

12. Al ejecutar el proyecto y consultar la vista que tiene la invocación al control tabla, se genera el siguiente html:



13. Para comprender el flujo es recomendable ejecutar el proyecto y poner un breakpoint en la invocación del control.

```
1  @using WebApp.Helpers;
2  <script src="~/Scripts/Views/vCustomers.js"></script>
3  @{
4      Layout = "~/Views/Shared/_Layout.cshtml";
5  }
6
7
8
9  @Html.CtrlTable("tblCustomers", title: "Customer List",
10     columnsTitle: "Id,Name,LastName,Age", ColumnsDataName: "Id,Name,LastName,Age")
```