Tipo : Guía de Enunciado
Capítulo : Servicios en Tiempo Real
Duración : 60 minutos

I. OBJETIVO

Instalación y configuración de Signal-R.

II. REQUISITOS

Los siguientes elementos de software son necesarios para la realización del laboratorio:

- Windows 10 (como mínimo Windows 8)
- Visual Studio 2017 (como mínimo Visual Studio 2015)

III. EJECUCIÓN DEL LABORATORIO

1. Abrir la solución del módulo anterior.

2. En el proyecto "**Cibertec.Mvc**" procedemos a crear una nueva carpeta con el nombre **Hubs**.

3. En esta carpeta cortamos y Pegamos el "**CustomerHub.cs**" y lo editamos con el siguiente código:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Web;
using Microsoft.AspNet.SignalR;

namespace Cibertec.Mvc
{
    public class CustomerHub : Hub
    {
        static List<int> CustomerIds = new List<int>();

        public void AddCustomerId(int id)
        {
            if(!CustomerIds.Contains(id)) CustomerIds.Add(id);
            Clients.All.customerStatus(CustomerIds);
        }

        public void RemoveCustomerId(int id)
        {
            if (CustomerIds.Contains(id)) CustomerIds.Remove(id);
            Clients.All.customerStatus(CustomerIds);
        }

        public override Task OnConnected()
        {
            return Clients.All.customerStatus(CustomerIds);
        }
    }
}
```

4. Dado que el mensaje a ver esta en nuestro helper de modal, procedemos a editar dicho helper que se encuentra en la carpeta "**App_Code**" y le agregamos el siguiente código:

```
@helper GetModal(string title, string closeFunction)
{
    <div class="modal fade" id="modal-container"
         tabindex="-1" role="dialog" aria-labelledby="myModalLabel">
        <div class="modal-dialog" role="document">
            <div class="modal-content">
                <div class="modal-header">
                    <button type="button" class="close"
                            data-dismiss="modal"
                            aria-label="Close">
                        <span aria-hidden="true">&times;</span>
                    </button>
                    <h4 class="modal-title">@title</h4>
                </div>
                <div class="modal-body">
                </div>
            </div>
        </div>
    </div>

    <script type="text/javascript">
        $(function () {
            $('#modal-container').on('hidden.bs.modal', function () {
                @closeFunction
            });
        });
    </script>
}
```

5. En la vista index de Customer, procedemos a actualizar la llamada a nuestro modal:

```
@Modal.GetModal("Customer", "customer.closeModal();")
```

```
17    <div class="pagination"></div>
18    <div class="content"></div>
19    <div class="pagination"></div>
20
21    @Modal.GetModal("Customer", "customer.closeModal();")
22
23    <script src="~/Scripts/jquery.bootpag.min.js"></script>
24    <script src="~/App/cibertec.js" type="text/javascript"></script>
25    <script src="~/App/customer.js" type="text/javascript"></script>
26    <script src="~/Scripts/jquery.signalR-2.2.2.js"></script>
27    <script src="/signalr/hubs"></script>
```

6. Modificaremos la vista modal "_**Edit.cshtml**" de la siguiente manera:

```
@model Cibertec.Models.Customer
<h2>Edit</h2>
@Messages.Message("inUse", "Edit", "danger", "This Customer is in use at
this moment.")

@using (Ajax.BeginForm("Edit", new { Controller = "Customer" },
                                    new AjaxOptions
                                    {
                                        HttpMethod = "POST",
                                        InsertionMode = InsertionMode.Replace,
                                        UpdateTargetId = "modal-body",
                                        OnSuccess =
"customer.success('edit');customer.removeCustomer(@Model.Id);"
                                    }, htmlAttributes: new { id = "editForm"
}))
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.Id)

        <div class="form-group">
            @Html.LabelFor(model => model.FirstName, htmlAttributes: new {
@class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.FirstName, new {
htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.FirstName, "",
new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.LastName, htmlAttributes: new {
@class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.LastName, new {
htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.LastName, "",
new { @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.City, htmlAttributes: new {
@class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.City, new { htmlAttributes
= new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.City, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Country, htmlAttributes: new {
@class = "control-label col-md-2" })
            <div class="col-md-10">
```

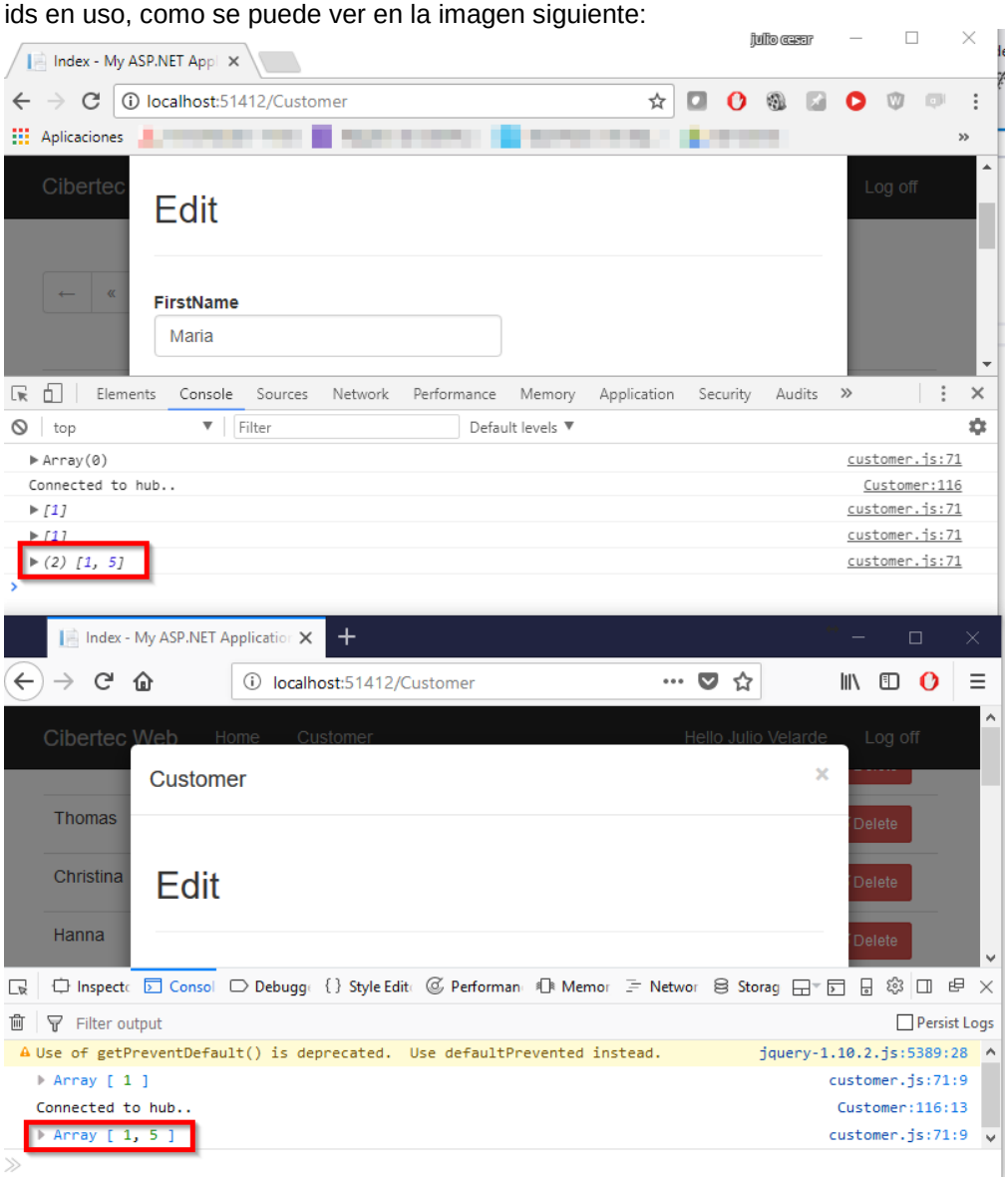
```
                @Html.EditorFor(model => model.Country, new {
htmlAttributes = new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Country, "", new
{ @class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Phone, htmlAttributes: new {
@class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Phone, new { htmlAttributes
= new { @class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Phone, "", new {
@class = "text-danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Save" class="btn btn-default"
/>
            </div>
        </div>
    </div>
}
<script type="text/javascript">
    $(function () {
        $.validator.unobtrusive.parse("#editForm");
        $('#inUse').addClass('hidden');
        customer.addCustomer(@Model.Id);
        customer.validate(@Model.Id);
    });

    customer.closeModal = function () {
        if(!customer.recordInUse)
            customer.removeCustomer(@Model.Id);
    };

</script>
```

7. Nos ubicamos en nuestra librería javascrip para el Customer, que se encuentra en la carpeta "**App**" y modificamos el código de la siguiente manera:

```
(function (customer) {
    customer.pages = 1;
    customer.rowSize = 25;
    customer.hub = {};
    customer.ids = [];
    customer.recordInUse = false;

    customer.success = successReload;
    customer.addCustomer = addCustomerId;
    customer.removeCustomer = removeCustomerId;
    customer.validate = validate;

    $(function () {
        connectToHub();
        init();
    });

    return customer;
```

```
        function successReload(option) {
            cibertec.closeModal(option);
        }

        function init() {
            $.get('/Customer/Count/' + customer.rowSize,
                function (data) {
                    customer.pages = data;
                    $('.pagination').bootpag({
                        total: customer.pages,
                        page: 1,
                        maxVisible: 5,
                        leaps: true,
                        firstLastUse: true,
                        first: '←',
                        last: '→',
                        wrapClass: 'pagination',
                        activeClass: 'active',
                        disabledClass: 'disabled',
                        nextClass: 'next',
                        prevClass: 'prev',
                        lastClass: 'last',
                        firstClass: 'first'
                    }).on('page', function (event, num) {
                        getCustomers(num);
                        });
                    getCustomers(1);
                });
        }
        function getCustomers(num) {
            var url = '/customer/List/' + num + '/' + customer.rowSize;
            $.get(url, function (data) {
                $('.content').html(data);
            });
        }
        function addCustomerId(id) {
            customer.hub.server.addCustomerId(id);
        }

        function removeCustomerId(id) {
            customer.hub.server.removeCustomerId(id);
        }

        function connectToHub() {
            customer.hub = $.connection.customerHub;
            customer.hub.client.customerStatus = customerStatus;
        }

        function customerStatus(customerIds) {
            console.log(customerIds);
            customer.ids= customerIds;
        }

        function validate(id) {
            customer.recordInUse = (customer.ids.indexOf(id) > -1);
            if (customer.recordInUse) {
                $('#inUse').removeClass('hidden');
            }
        }

    })(window.customer = window.customer || {});
```

8. Compilamos y podremos ver en la consola del navegador que se van incrementando los ids en uso, como se puede ver en la imagen siguiente:



IV. EVALUACIÓN

1. ¿Qué son los Websockets?
2. ¿Qué es un HUB de SignalR?
3. ¿Por qué se prefiere usar como protocolo de transporte Websockets?
4. ¿Cuál es la diferencia más importante entre SignalR y SignalR Core?