# NortwindDBJSON.ADO Project App Notes

The NorthwindDBJSON.ADO project is created using the ASP.NET Core 3.1 Web Application template to access customer-order data in a SQL Server database.

There are three key components within this project:

- <hostname>/api/CustomerOrder
- <hostname>/Customers/index
- <hostname>/CustomerOrderJson.html

## <hostname>/api/CustomerOrder

The "<hostname>/api/CustomerOrder" URL is accessing an ASP.NET Core Web API, which returns customer-order data in JSON format.

## <hostname>/Customers/index

The "<hostname>/Customers/index" URL is accessing Index() IActionResult, in CustomersController, to retrieve and display customer data, using the "\Views\Customers\index.cshtml" view, in the Customers controller

The "\Views\Customers\index.cshtml" view is part of the codes scaffolded by the "new controller wizard" using the "MVC Controller with views, using Entity Framework" scaffolding extension, which generated the following CRUD Views:

- \Views\Customers\Create.cshtml
- \Views\Customers\Delete.cshtml
- \Views\Customers\Details.cshtml
- \Views\Customers\Edit.cshtml
- \Views\Customers\index.cshtml

## <hostname>/CustomerOrderJson.html

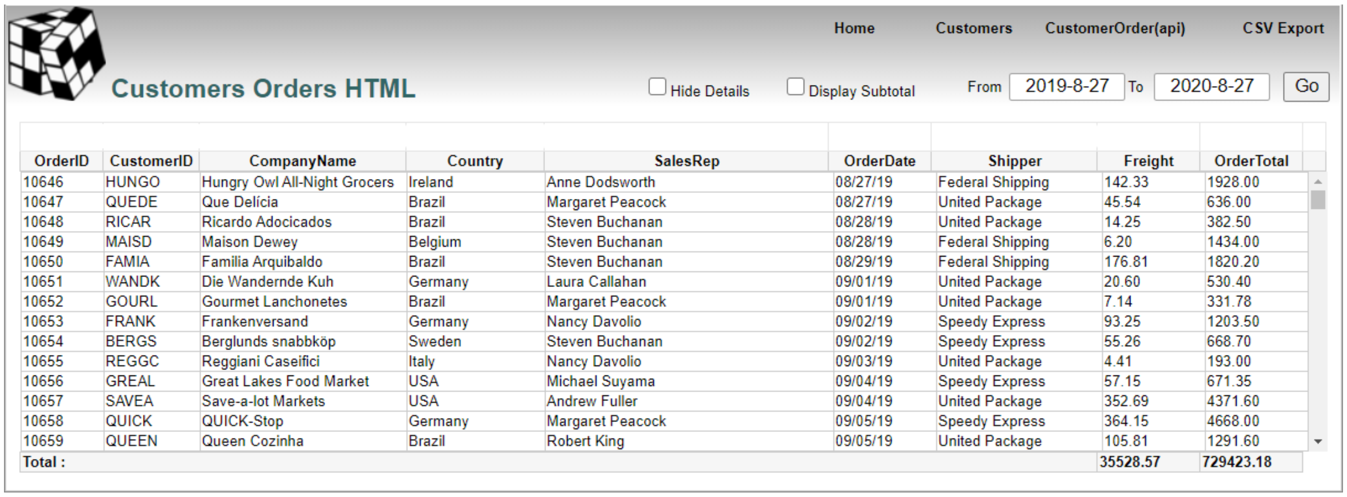
The "<hostname>/CustomerOrderJson.html" web application is simply an HTML page with JavaScript and CSS (no framework in use) to access the back-end SQL database via a Web API, which return data in JSON format.

There is an additional JavaScript file in the "\NorthwindDBJSON\NorthwindDBJSON.ADO\wwwroot\js" folder:

- The Utility.js file contains some helper functions.

The **<hostname>/api/CustomerOrder** URL is the WebAPI created to access back-end SQL database and provide the data in JSON format.

A working app is deployed to Azure and accessible via the following URL:
https://jsmeetup.azurewebsites.net/CustomerOrderJson.html



| OrderID | CustomerID | CompanyName | Country | SalesRep | OrderDate | Shipper | Freight | OrderTotal |
|---|---|---|---|---|---|---|---|---|
| 10646 | HUNGO | Hungry Owl All-Night Grocers | Ireland | Anne Dodsworth | 08/27/19 | Federal Shipping | 142.33 | 1928.00 |
| 10647 | QUEDE | Que Delícia | Brazil | Margaret Peacock | 08/27/19 | United Package | 45.54 | 636.00 |
| 10648 | RICAR | Ricardo Adocicados | Brazil | Steven Buchanan | 08/28/19 | United Package | 14.25 | 382.50 |
| 10649 | MAISD | Maison Dewey | Belgium | Steven Buchanan | 08/28/19 | Federal Shipping | 6.20 | 1434.00 |
| 10650 | FAMIA | Familia Arquibaldo | Brazil | Steven Buchanan | 08/29/19 | Federal Shipping | 176.81 | 1820.20 |
| 10651 | WANDK | Die Wandernde Kuh | Germany | Laura Callahan | 09/01/19 | United Package | 20.60 | 530.40 |
| 10652 | GOURL | Gourmet Lanchonetes | Brazil | Margaret Peacock | 09/01/19 | United Package | 7.14 | 331.78 |
| 10653 | FRANK | Frankenversand | Germany | Nancy Davolio | 09/02/19 | Speedy Express | 93.25 | 1203.50 |
| 10654 | BERGS | Berglunds snabbköp | Sweden | Steven Buchanan | 09/02/19 | Speedy Express | 55.26 | 668.70 |
| 10655 | REGGC | Reggiani Caseifici | Italy | Nancy Davolio | 09/03/19 | United Package | 4.41 | 193.00 |
| 10656 | GREAL | Great Lakes Food Market | USA | Michael Suyama | 09/04/19 | Speedy Express | 57.15 | 671.35 |
| 10657 | SAVEA | Save-a-lot Markets | USA | Andrew Fuller | 09/04/19 | United Package | 352.69 | 4371.60 |
| 10658 | QUICK | QUICK-Stop | Germany | Margaret Peacock | 09/05/19 | Speedy Express | 364.15 | 4668.00 |
| 10659 | QUEEN | Queen Cozinha | Brazil | Robert King | 09/05/19 | United Package | 105.81 | 1291.60 |
| **Total :** | | | | | | | **35528.57** | **729423.18** |

By clicking on each column's heading, data display can be sort in ascending or descending order based on the clicked-heading.

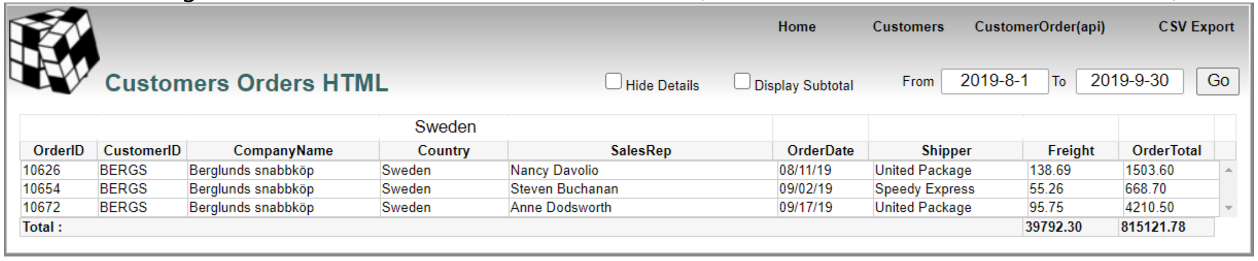
By enabling the "Display Subtotal" checkbox, then clicking on a data column's heading will generate subtotal based on the clicked-heading, such as:

- Clicking on "Country", the display will group and generate subtotal for each country.
- Clicking on "CustomerID", the display will group and generate subtotal for each customer.
- Clicking on "SalesRep", the display will group and generate subtotal for each SalesRep.

To view only the subtotal without the details, select the "Hide Details" checkbox.
(Hiding details only work when "Display Subtotal" is enabled)

## Filtering Data

Data can be filtered and limited within the specified From and To dates. In addition, the text-box above the column-heading can be used to enter a value to filter data (not all column can be used to filter data).



From 2019-8-1 To 2019-9-30

| | | | Sweden | | | | | |
|---|---|---|---|---|---|---|---|---|
| OrderID | CustomerID | CompanyName | Country | SalesRep | OrderDate | Shipper | Freight | OrderTotal |
| 10626 | BERGS | Berglunds snabbköp | Sweden | Nancy Davolio | 08/11/19 | United Package | 138.69 | 1503.60 |
| 10654 | BERGS | Berglunds snabbköp | Sweden | Steven Buchanan | 09/02/19 | Speedy Express | 55.26 | 668.70 |
| 10672 | BERGS | Berglunds snabbköp | Sweden | Anne Dodsworth | 09/17/19 | United Package | 95.75 | 4210.50 |
| Total : | | | | | | | 39792.30 | 815121.78 |

Data filtering example:

- Above the "Country" column-heading, enter "SWEDEN" and press enter will filter and display data for Sweden.
- Change the From-date to "2019-8-1" and To-date to "2019-9-30' will further filter the data and only show orders between these two days

| **Note:** |
|---|
| Click on the "Go" button to process data filtering.<br>When entering filter value in the column-heading, the enter key will trigger data filtering. If not, click on the "Go" button. |

## CSV Export

The "CSV export" will generate a comma-delimited CSV file based on the data currently showing on the web page.

## <hostname>/CustomerOrderJson.html

The "<hostname>/CustomerOrderJson.html" web application is simply an HTML page with JavaScript and CSS (no framework in use) to access the back-end SQL database via a Web API, which return data in JSON format.

There is an additional JavaScript file in the "\NorthwindDBJSON\NorthwindDBJSON.ADO\wwwroot\js" folder:

- The Utility.js file contains some helper functions.

In the CustomerOrderJson.js JavaScript file, the following block of code provides the mechanism to execute the intended JavaScript when the HTML content is loaded:

```
let PageMngr = null;

window.onload = window_onload;
function window_onload()
{
    try
    {
        PageMngr = new PageManager();
        PageMngr.DisplayInit();
        PageMngr.GetJsonData();
    }
    catch (ex)
    {
        alert(ex.description);
        window.status = ex.description;
    }
}
```

The "PageMngr.DisplayInit()" calls the following function to initialize display:

```
    this.DisplayInit = DisplayInit;
    function DisplayInit()
    {
        let _today = new Date();
        ToDate = _today.getFullYear() + "-" + (_today.getMonth() + 1) + "-" + _today.getDate();
        if (txtToDate) { txtToDate.value = ToDate;}

        let _fromDate = new Date();
        _fromDate.setMonth(_fromDate.getMonth() - 12);
        FromDate = _fromDate.getFullYear() + "-" + (_fromDate.getMonth() + 1) + "-" + _fromDate.getDate();
        if (txtFromDate) { txtFromDate.value = FromDate;}

        DisplaySubtotal = "0";
        chkDisplaySubTotals.checked = false;

        HideDetails = "0";
        chkHideDetails.checked = false;
    }
```

The "PageMngr.GetJsonData()" calls the following function, which uses XMLHttpRequest to retrieve data from a JSON WebAPI:

```
this.GetJsonData = GetJsonData;
function GetJsonData()
{
    let queryString = "?FromDate=" + FromDate + "&ToDate=" + ToDate;

    if (OrderBy.trim() != "") { queryString += "&OrderBy=" + OrderBy; }
    if (AscDesc.trim() != "")
    { queryString += "&AscDesc=" + AscDesc; }
    else
    { queryString += "&AscDesc=0"; }
    if (OrderID.trim() != "") { queryString += "&OrderID=" + OrderID; }
    if (CustomerID.trim() != "") { queryString += "&CustomerID=" + CustomerID; }
    if (CompanyName.trim() != "") { queryString += "&CompanyName=" + CompanyName; }
    if (Country.trim() != "") { queryString += "&Country=" + Country; }
    if (SalesRep.trim() != "") { queryString += "&SalesRep=" + SalesRep; }
    if (Shipper.trim() != "") { queryString += "&Shipper=" + Shipper; }

    let XHR = new XMLHttpRequest();
    XHR.open("GET", apiURL + queryString, true);
    XHR.timeout = 10000;  // in milliseconds
    XHR.setRequestHeader('Access-Control-Allow-Headers', '*');
    XHR.send();

    XHR.onload = function ()
    {
        DisplayJsonData(XHR.responseText);
    }

    XHR.onerror = function ()
    {
        // routine to handle error
        alert("XMLHttpRequest Error");
    }

    XHR.ontimeout = function ()
    {
        // routine to handle timeout
        alert("XMLHttpRequest TimeOut");
    }

}
```

After retrieving JSON data from the WebAPI, the following function is called to display data to an HTML table:

```
let ReportBody = document.getElementById("ReportBody");
let ReportFooterBody = document.getElementById("ReportFooterBody");
function DisplayJsonData(_response)
{
    let _displayData = "";
    let _orderByValue = "";
    let _subTotalOrder = 0.0;
    let _subTotalFreight = 0.0;

    let _jsonData = JSON.parse(_response);

    if (_jsonData.length > 0)
    {
        _orderByValue = _jsonData[0]["orderByValue"];
    }

    for (let i = 0; i < _jsonData.length; i++)
    {
        if (OrderBy != "" && chkDisplaySubTotals.checked)
        {
            if (_orderByValue == _jsonData[i]["orderByValue"])
            {
                _subTotalOrder += parseFloat(_jsonData[i]["orderTotal"].replace(/,/g, ''));
                _subTotalFreight += parseFloat(_jsonData[i]["freight"].replace(/,/g, ''));
            }
            else
            {
                _displayData +=
                      "<tr class='rwSubtotal'><th colspan='7'>Subtotal: ( " + _orderByValue + " ): </th>"
                    + "<th>" + _subTotalFreight.toFixed(2) + "</th>"
                    + "<th>" + _subTotalOrder.toFixed(2) + "</th></tr>";

                _subTotalOrder = parseFloat(_jsonData[i]["orderTotal"].replace(/,/g, ''));
                _subTotalFreight = parseFloat(_jsonData[i]["freight"].replace(/,/g, ''));
                _orderByValue = _jsonData[i]["orderByValue"];
            }
        }

        let _orderTotal = parseFloat(_jsonData[i]["orderTotal"].replace(/,/g, ''));
        let _freight = parseFloat(_jsonData[i]["freight"].replace(/,/g, ''));
        if ((!chkHideDetails.checked) || (!chkDisplaySubTotals.checked))
        {
            _displayData += "<tr><td>" + _jsonData[i]["orderID"] + "</td>"
                + "<td>" + _jsonData[i]["customerID"] + "</td>"
                + "<td>" + _jsonData[i]["companyName"] + "</td>"
                + "<td>" + _jsonData[i]["country"] + "</td>"
                + "<td>" + _jsonData[i]["salesRep"] + "</td>"
                + "<td>" + _jsonData[i]["orderDate"] + "</td>"
                + "<td>" + _jsonData[i]["shipper"] + "</td>"
                + "<td>" + _freight.toFixed(2) + "</td>"
                + "<td>" + _orderTotal.toFixed(2) + "</td></tr>";
        }

        TotalOrder += _orderTotal;
        TotalFreight += _freight;

    }

    if (OrderBy != "" && chkDisplaySubTotals.checked)
    {
        _displayData += "<tr class='rwSubtotal'><th colspan='7'>Subtotal: ( " + _orderByValue + " ): </th>"
            + "<th>" + _subTotalFreight.toFixed(2) + "</th>"
            + "<th>" + _subTotalOrder.toFixed(2) + "</th></tr>";
    }

    if (ReportBody)
    {
        ReportBody.innerHTML = _displayData;
    }

    if (ReportFooterBody)
    {
        let _footerData = "<tr class='rwFooter'><th>Total :</th>"
            + "<th>" + TotalFreight.toFixed(2) + "</th>"
            + "<th>" + TotalOrder.toFixed(2) + "</th></tr>";

        ReportFooterBody.innerHTML = _footerData;
    }

}
```

## Mouse Events

Mouse events are captured in different region of the web application, process by the following functions:

The following function process mouse events in the Form Header region:

```
    let tblFormHeader = document.getElementById("tblFormHeader");
    if (tblFormHeader) { tblFormHeader.onclick = tblFormHeader_onclick; }
    function tblFormHeader_onclick(e)
    {
        let evt = e || window.event;
        let elm = evt.srcElement || evt.target;
        if ((elm.id == "chkDisplaySubTotals") || (elm.id == "chkHideDetails"))
        {
            RefreshDisplay();
        }
    }
```

The following function process mouse events in the data-column heading region:

```
    let tblHeader = document.getElementById("tblReportHeader");
    if (tblHeader) { tblHeader.onclick = tblHeader_onclick; }
    function tblHeader_onclick(e)
    {
        let evt = e || window.event;
        let elm = evt.srcElement || evt.target;
        if (elm.tagName == "A")
        {
            let OrderBy = elm.id.replace("OrderBy.", "");
            if (hdnOrderBy.value == OrderBy)
            {
                hdnAscDesc.value = (hdnAscDesc.value == "1") ? "" : "1";
            }
            else
            {
                hdnAscDesc.value = "";
                hdnOrderBy.value = OrderBy;
            }
            RefreshDisplay();
            return false;
        }
        return true;
    }
```

The following function process mouse and keyboard events in the data-column heading 's filter region:

```
    let Filters = document.getElementById("Filters");
    if (Filters) { Filters.onkeypress = txt_keypress; }
    function txt_keypress(e)
    {
        let evt = e || window.event;
        let kc = evt.keyCode || evt.which;
        if (kc == 13)
        {
            RefreshDisplay();
            return false;
        }
        return true;
    }
```

All mouse/keyboard event handler call the following function to refresh display:

```
function RefreshDisplay()
{
    let elms = document.getElementsByTagName("INPUT");
    for (let ix = 0; ix < elms.length; ix++)
    {
        let elm = elms[ix];
        if (elm.type == "text")
        {
            switch (elm.id)
            {
                case "txtFromDate":
                    FromDate = elm.value;
                    break;
                case "txtToDate":
                    ToDate = elm.value;
                    break;
                case "txtOrderID":
                    OrderID = elm.value;
                    break;
                case "txtCustomerID":
                    CustomerID = elm.value;
                    break;
                case "txtCompanyName":
                    CompanyName = elm.value;
                    break;
                case "txtCountry":
                    Country = elm.value;
                    break;
                case "txtSalesRep":
                    SalesRep = elm.value;
                    break;
                case "txtShipper":
                    Shipper = elm.value;
                    break;
            }
        }
        else if (elm.type == "hidden")
        {
            switch (elm.id)
            {
                case "hdnOrderBy":
                    OrderBy = elm.value;
                    break;
                case "hdnAscDesc":
                    AscDesc = elm.value;
                    break;
            }
        }
        else if (elm.type == "checkbox")
        {
            switch (elm.id)
            {
                case "chkDisplaySubTotals":
                    DisplaySubtotal = elm.checked ? "1" : "";
                    break;
                case "chkHideDetails":
                    HideDetails = elm.checked ? "1" : "";
                    break;
            }
        }
    }

    GetJsonData();

}
```