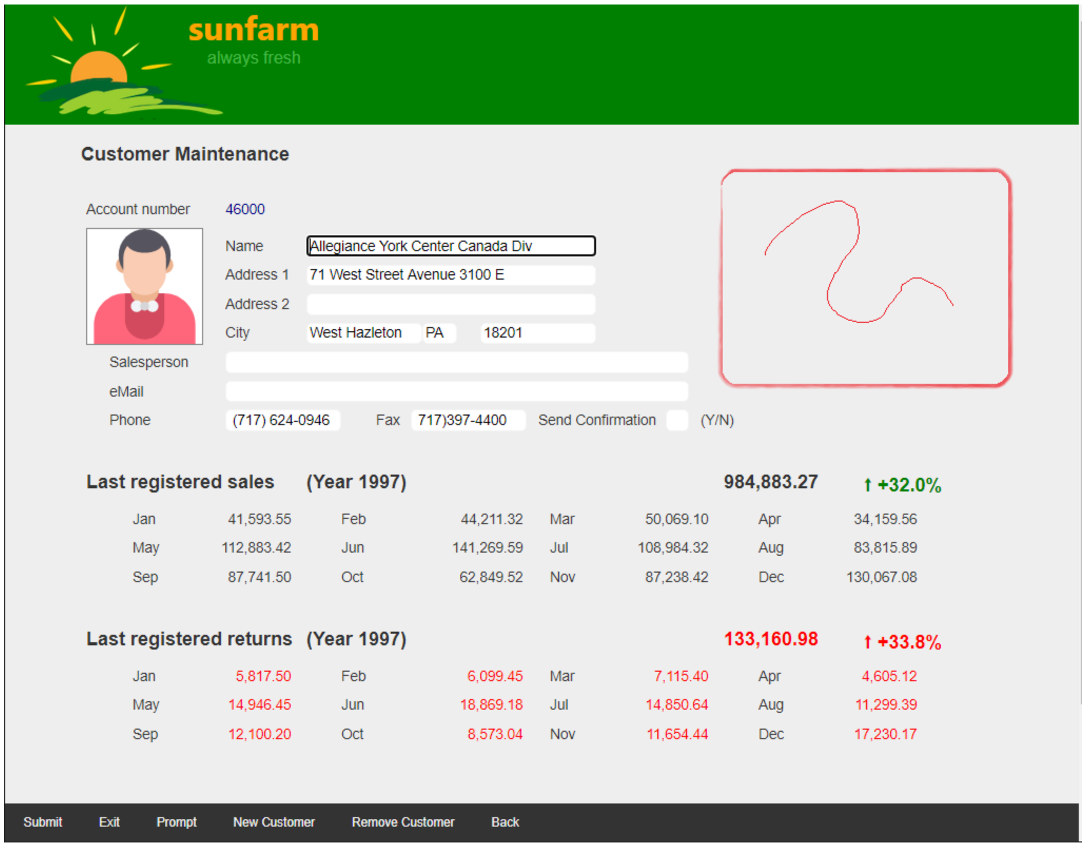
Adding a 3rd Party Chart to a Page

You may have noticed that when reorganizing the data on the Page on the previous Part 2, we left a distinctive area on the upper-right hand corder of the page:



Customer Maintenance

Account number	46000
Name	Allegiance York Center Canada Div
Address 1	71 West Street Avenue 3100 E
Address 2	
City	West Hazleton PA 18201
Salesperson	
eMail	
Phone	(717) 624-0946
Fax	717)397-4400
Send Confirmation	(Y/N)

Last registered sales (Year 1997) 984,883.27 ↑ +32.0%

Jan	41,593.55	Feb	44,211.32	Mar	50,069.10	Apr	34,159.56
May	112,883.42	Jun	141,269.59	Jul	108,984.32	Aug	83,815.89
Sep	87,741.50	Oct	62,849.52	Nov	87,238.42	Dec	130,067.08

Last registered returns (Year 1997) 133,160.98 ↑ +33.8%

Jan	5,817.50	Feb	6,099.45	Mar	7,115.40	Apr	4,605.12
May	14,946.45	Jun	18,869.18	Jul	14,850.64	Aug	11,299.39
Sep	12,100.20	Oct	8,573.04	Nov	11,654.44	Dec	17,230.17

We want to display two pieces of data:

1. The Customer Status (“Active”, “Closed”, “Over limit”, “Refer” and “Suspended”) at the top of that box, and
2. A Sales Chart that makes it easier to understand the Sales performance trend.

Green-screen cryptic codes may now be displayed descriptively

The original green-screen page would show:

```
Status: A (F4)
```

“A” code is too cryptic, only experienced operators would remember that it means “Active”. Without changing the Business Logic, we can improve the presentation of that information by desplaying full descriptions: we have plenty of space now!

While changing the presentation of the “status” we do not want to loose the prompting ability for that field. To avoid affecting too much the Business Logic, let’s work on a presentation trick.

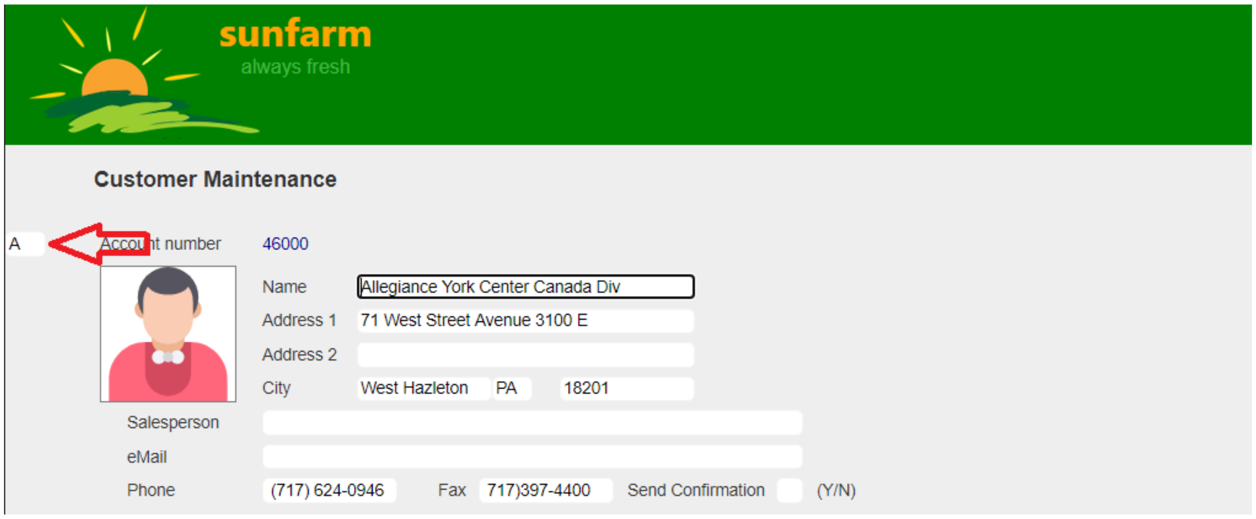
First, put back the CUSTREC.SFSTATUS field back on the Page. For convenience, show it at column position “1”.

Markup for Row “2”:

```
<div Row="2">
    <DdsConstant Col="8" Text="Account number" />
    <DdsDecField class="left-aligned-field" Col="20" For="CUSTREC.SFCUSTNO"
       VirtualRowCol="5,27" Color="DarkBlue" EditCode="Z" Comment="CUSTOMER NUMBER" />
    <DdsCharField Col="1" For="CUSTREC.SFSTATUS" VirtualRowCol="15,27" PositionCursor="44"/>
</div>
```

Run the Website Application (no need to recompile business logic):

You can still get to the Status code:



Customer Maintenance

A

Account number 46000

Name Allegiance York Center Canada Div

Address 1 71 West Street Avenue 3100 E

Address 2

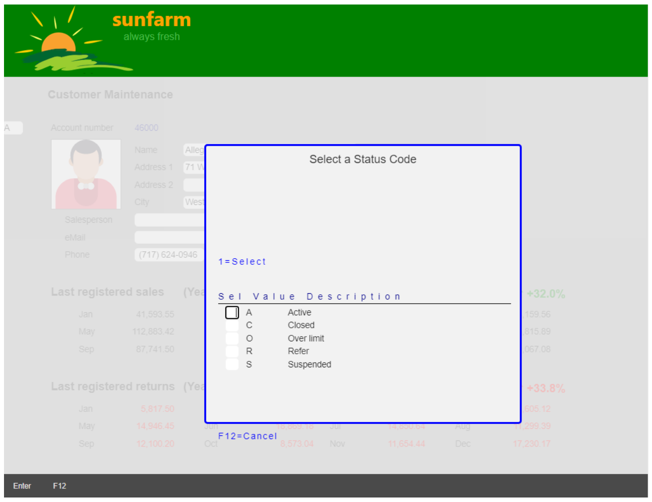
City West Hazleton PA 18201

Salesperson

eMail

Phone (717) 624-0946 Fax 717)397-4400 Send Confirmation (Y/N)

And prompt to change its value:



Select a Status Code

1=Select

Sel	Value	Description
☐	A	Active
	C	Closed
	O	Over limit
	R	Refer
	S	Suspended

F12=Cancel

Now let’s add a standard HTML <span> element to display the Status description on a proper place:

```
<div Row="2">
    <DdsConstant Col="8" Text="Account number" />
    <DdsDecField class="left-aligned-field" Col="20" For="CUSTREC.SFCUSTNO" VirtualRowCol="5,27"
Color="DarkBlue" EditCode="Z" Comment="CUSTOMER NUMBER" />
    <DdsCharField Col="1" For="CUSTREC.SFSTATUS" VirtualRowCol="15,27" PositionCursor="44"
tabIndex=10 />
    <span class="box-highlight-center-field" ExpoGridCol="72/90">@Model.CUSTREC.SF_STATUS_NAME</
span>
</div>
```

You may recognize that CSS Style box-highlight-center-field, has not been defined, and you are right, here it is (you can add it to site.css file):

```
.box-highlight-center-field {
     background-color: black;
     color: white;
     text-align: center;
}
```

Nothing fancy, a white on black box with the text centered.

What is @Model.CUSTREC.SF_STATUS_NAME?

We have been using For=“” to reference fields in attributes for Expo tag-helpers, but when we use standard HTML elements, where we want to refer to the content of an HTML element we need to let Razor out intent. Using @ symbol will let use switch to C# code, were we can refer by contents on symbols defined on the Model. @Model.CUSTREC will get us to the instance of the CUSTREC DdsRecord, but field SF_STATUS_NAME does not exist.

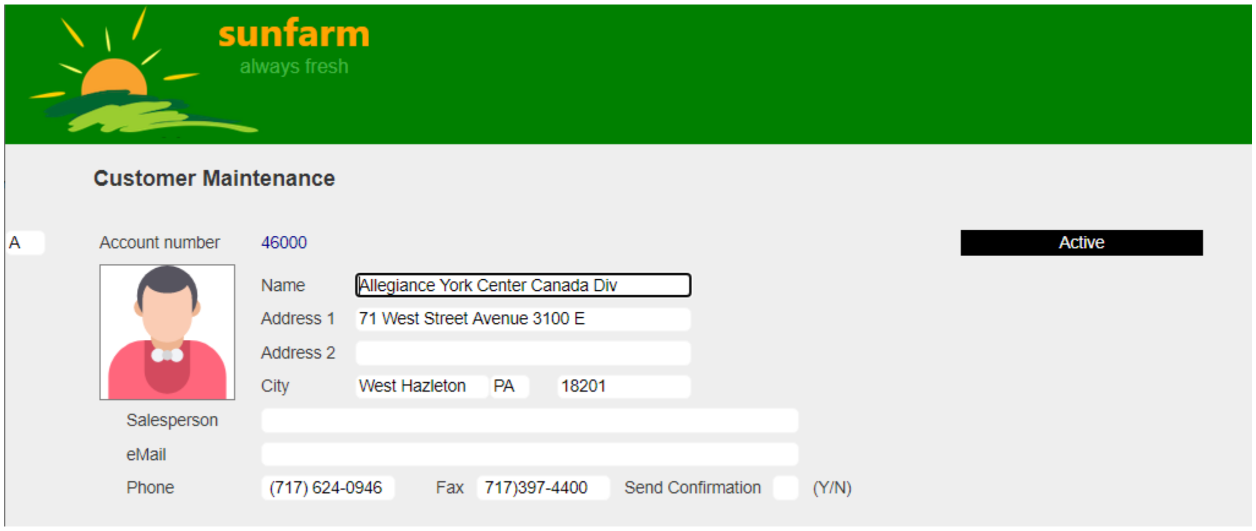
We can fix that, by defining it this way:

…\CustomerAppSite\Areas\CustomerAppViews\Pages\CUSTDSPF.cshtml.cs:

```
    [Char(1)]
    public string SFSTATUS { get; set; }

    public string SF_STATUS_NAME { get { return statusCodeToName(); } }

    private string statusCodeToName()
    {
        switch (SFSTATUS)
        {
            case "A":
                return "Active";
            case "C":
                return "Closed";
            case "O":
                return "Over Limit";
            case "R":
                return "Refer";
            case "S":
                return "Suspended";
        }

        return "(Undefined Status)";
    }
```

Notice how we don't qualify the new properties with [Char(nn)] because we are referring to C# standard string properties — additionally we don't want to add it to the Workstation DataSet— This is just presentation logic.

Run the Website Application (and make sure the CSS cache is refreshed), and you should see the following output.[1]



Customer Maintenance

A Account number 46000 Active

Name Allegiance York Center Canada Div
Address 1 71 West Street Avenue 3100 E
Address 2
City West Hazleton PA 18201
Salesperson
eMail
Phone (717) 624-0946 Fax 717)397-4400 Send Confirmation (Y/N)

[1] Commit: "Descriptive Customer Status field".

We can trick the user by overlapping the original Status code field on top of the descriptive new presentation field, and make it invisible.

Let's declare a new CSS Style:

```
$\SunFarm\CustomerAppSite\wwwroot\css\site.css

.present-no-visible-field {
     opacity:0.1;
}
```

And change our Row="2" such that both the original Status code and the new Status Descriptions are written to the page, but where the Status code is not visible:

```
...\CustomerAppSite\Areas\CustomerAppViews\Pages\CUSTDSPF.cshtml
```

```
<div Row="2">
    <DdsConstant Col="8" Text="Account number" />
    <DdsDecField class="left-aligned-field" Col="20" For="CUSTREC.SFCUSTNO"
      VirtualRowCol="5,27" Color="DarkBlue" EditCode="Z" Comment="CUSTOMER NUMBER" />
    <span class="box-highlight-center-field" ExpoGridCol="72/90">@Model.CUSTREC.SF_STATUS_NAME</
span>
    <DdsCharField class="present-no-visible-field" Col="72" ColSpan="18" For="CUSTREC.SFSTATUS"
VirtualRowCol="15,27" PositionCursor="44" />
</div>
```

Note that order is important. When overlapping elements and one is input capable, the later must be on top of the former. Browser renders elemements in the order in which they are described by HTML, in this case we want "CUSTREC.SFSTATUS" render on top of @Model.CUSTREC.SF_STATUS_NAME

Run the Website Application and you can verify that the Status Description may still be "clickable" and prompting to change its value still works.

2

[2] Commit "Chart heading showing the Status Description Centered".

Chart Placeholder

They Chart will be placed vertically at the same level as Row="3" on a box with width = 300 pixels and height: 200 piles. The box will have a light colored gray border.

We can start with standard DIV HTML element. We will position it like we have been positioning other elements before, under s particular Row with a horizontal displacement indicated by a Column attribute.

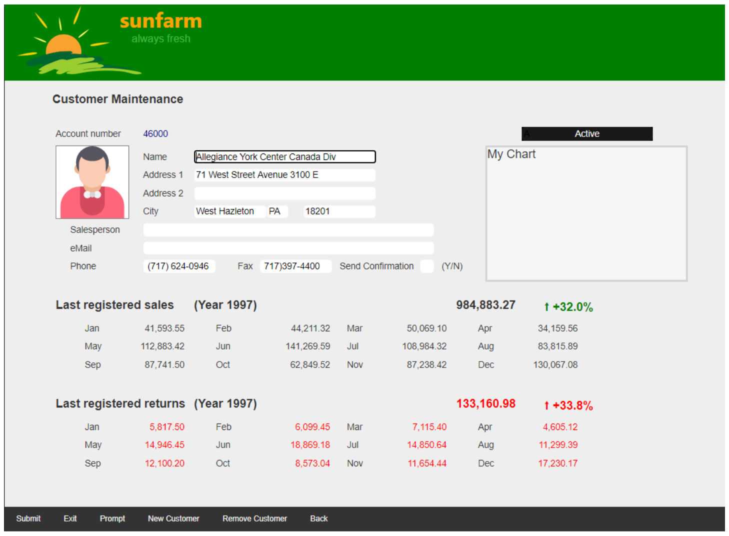
The starting Markup is:

```
<div Row="3">
    <img id="customer-icon" ExpoCol="8" src="~/customer-icon.svg" />
    <div id="custrec-chart" ExpoCol="67">My Chart</div>
</div>
```

With the custrec-chart style defined in CSS:

```
$\SunFarm\CustomerAppSite\wwwroot\css\site.css


#custrec-chart {
    position: absolute;
    background-color: whitesmoke;
    width: 300px;
    height: 200px;
    border-style: solid;
    border-color: lightgrey;
    border-width: medium;
}
```

Such element shows like this:[3]



Customer Maintenance

Account number 46000

Active

Name: Allegiance York Center Canada Div
Address 1: 71 West Street Avenue 3100 E
Address 2:
City: West Hazleton PA 18201
Salesperson:
eMail:
Phone: (717) 624-0946 Fax 717)397-4400 Send Confirmation (Y/N)

My Chart

Last registered sales (Year 1997) **984,883.27** **↑ +32.0%**

Jan	41,593.55	Feb	44,211.32	Mar	50,069.10	Apr	34,159.56
May	112,883.42	Jun	141,269.59	Jul	108,984.32	Aug	83,815.89
Sep	87,741.50	Oct	62,849.52	Nov	87,238.42	Dec	130,067.08

Last registered returns (Year 1997) **133,160.98** **↑ +33.8%**

Jan	5,817.50	Feb	6,099.45	Mar	7,115.40	Apr	4,605.12
May	14,946.45	Jun	18,869.18	Jul	14,850.64	Aug	11,299.39
Sep	12,100.20	Oct	8,573.04	Nov	11,654.44	Dec	17,230.17

[3] Commit: "Chart Placeholder"

We will be using AM**CHARTS** Version 4 third party JavaScript Engine.

You may want to visit: https://www.amcharts.com but for the purposes of this Guide we will present in the following paragraphs, the basic usage of that product.

The Chart is we want is an XY line chart that will plot one value for every one of the twelve months in a given year, and connect those points with lines. The effect is to visually see if the trend is to grow or reduce sales number month by month, and a sharp for the year.

We will not spend too much time in detailing the chart (which requires deeper understanding of the AM**CHARTS** API.

Basic Chart Terminology:

1. Chart type: XYChart
2. A Series is a collection of data points.
3. Category X data is ***months***.
4. Category Y values is ***sales***.
5. Value Title is “Sales”

The AM**CHARTS** is fed by JavaScript, which we will add at the end of our CUSTDSPF.chsml file.

The Script algorithm is:

1. Find HTML **div** placeholder element by ID
2. If the element exists, we can fairly assume that the CUSTREC record is active on display, therefore we will create the Chart.
3. Establish data-fields Category items as “month”
4. Establish Value axis as “sales”
5. Establish Value series as collection of fields with have month, sales point data.
6. Provide the chart data in JSON format

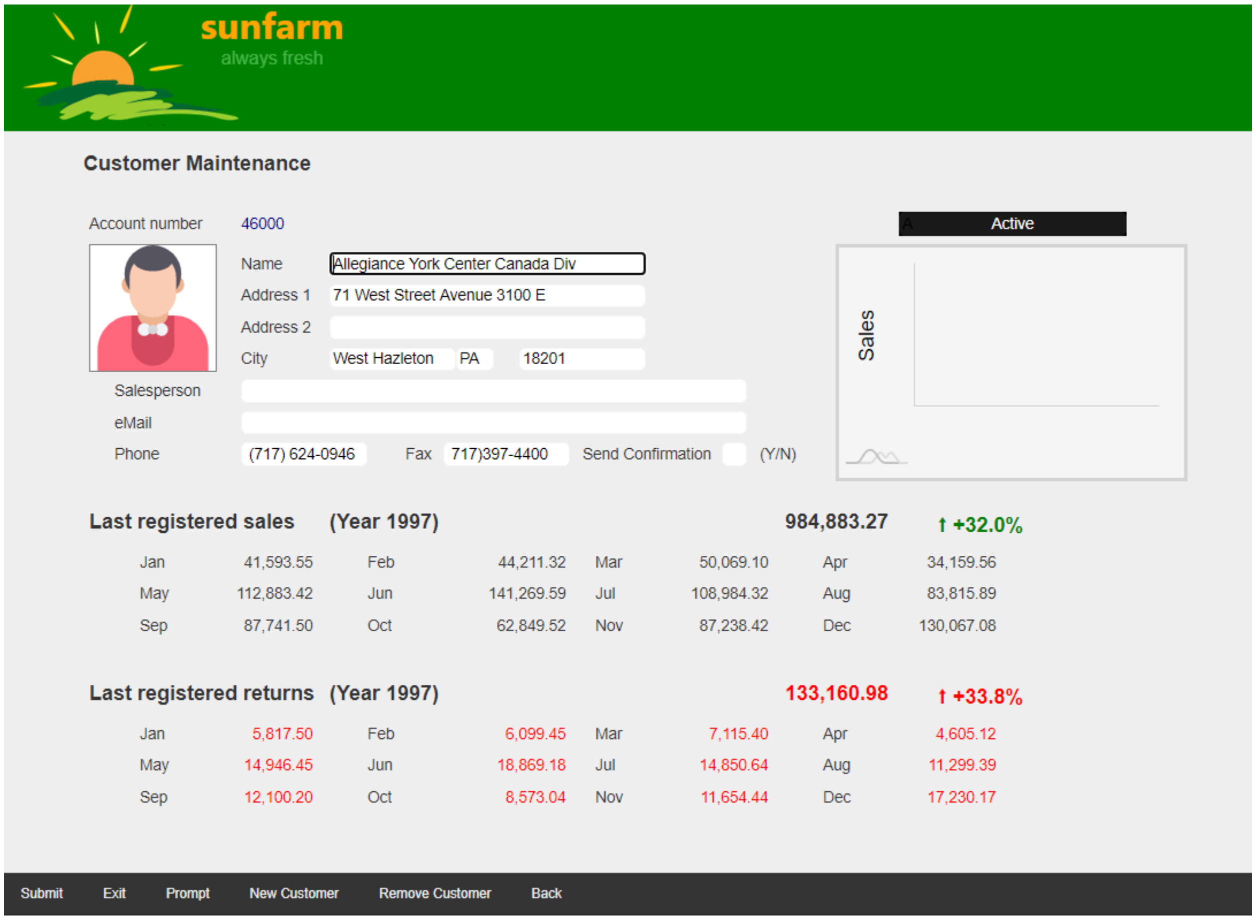
Append the following script(s) at the end of the Markup file:

```
<script src="https://cdn.amcharts.com/lib/4/core.js"></script>
<script src="https://cdn.amcharts.com/lib/4/charts.js"></script>
<script>
    const CHART_ID = 'custrec-chart';
    const chartEl = document.getElementById(CHART_ID);

    if (chartEl !== null) {
        let salesJsonData = chartEl.innerHTML;
        let chart = am4core.create(CHART_ID, am4charts.XYChart);
        let categoryAxis = chart.xAxes.push(new am4charts.CategoryAxis());
        categoryAxis.dataFields.category = "month";

        let valueAxis = chart.yAxes.push(new am4charts.ValueAxis());
        valueAxis.title.text = "Sales";

        let series = chart.series.push(new am4charts.LineSeries());
        series.name = "Sales";
        series.stroke = am4core.color("#CDA2AB");
        series.strokeWidth = 3;
        series.dataFields.valueY = "sales";
        series.dataFields.categoryX = “month";
        const chartData = JSON.parse(salesJsonData);
        chart.data = chartData.data;
    }
</script>
```

The two-first script lines, refer to the cdn address for the AM**CHARTS** production JavaScript. Note that the AM**CHARTS** library allows free usage, with the penalty to show a watermark on the chart, —which obscures the output, but is reasonable for a prototype—

The third script our custom Algorithm described above.

If you run the Website Application now, you will see:[4]



Customer Maintenance

Account number	46000
Name	Allegiance York Center Canada Div
Address 1	71 West Street Avenue 3100 E
Address 2	
City	West Hazleton PA 18201
Salesperson	
eMail	
Phone	(717) 624-0946
Fax	717)397-4400
Send Confirmation	(Y/N)

Last registered sales (Year 1997) 984,883.27 ↑ +32.0%

Jan	41,593.55	Feb	44,211.32	Mar	50,069.10	Apr	34,159.56
May	112,883.42	Jun	141,269.59	Jul	108,984.32	Aug	83,815.89
Sep	87,741.50	Oct	62,849.52	Nov	87,238.42	Dec	130,067.08

Last registered returns (Year 1997) 133,160.98 ↑ +33.8%

Jan	5,817.50	Feb	6,099.45	Mar	7,115.40	Apr	4,605.12
May	14,946.45	Jun	18,869.18	Jul	14,850.64	Aug	11,299.39
Sep	12,100.20	Oct	8,573.04	Nov	11,654.44	Dec	17,230.17

Note that the Chart is empty (we have not provided valid JSON data points). Also note that we got out "free" penalty, the AM**CHARTS** watermark logo.

The plumbing of the Chart is now in place, all we need to do is set as the DIV inner html, the data in JSON format.

---

[4] Commit "Empty Chart"

Producing the Chart data in JSON format

Change the Markup for Row="3", such that the contents of our chart placeholder has a reference to a new property on out server-side Model:

```
<div Row="3">
    <img id="customer-icon" ExpoCol="8" src="~/customer-icon.svg" />
    <div id="custrec-chart" ExpoCol="67">@Model.CUSTREC.SALES_CHART_DATA</div>
</div>
```

The script we provided to drive AM**CHARTS** has the following code:

```
        let salesJsonData = chartEl.innerHTML;
.
.
.
        const chartData = JSON.parse(salesJsonData);
        chart.data = chartData.data;
```

You can read this as:
1. Extract the HTML string content from our chart HTML div element (our placeholder).
2. Using the DOM JSON object, parse the string text and produce a JavaScript object, keeping a reference in the constant **chartData**.
3. **chartData** has a property called "data" with the data points expected by AM**CHARTS** such that rendering can be executed.

Implementing SALES_CHART_DATA getter property

Let's add the following erat-only property on our CUSREC Model:

```
[Char(20)]
public string PERCENT_CHANGE_RETURNS { get; private set; }
.
.
.
public string SALES_CHART_DATA { get { return FormatChartData(); } }
```

Note that the type is C# ***string*** type.

The getter method calls FormatChartData. We can safely assume that when SALES_CHART_DATA getter is executed, the Sales properties (CSSALES01 thru CSSALES12) have been populated with current values.

Add the following implementation for FormatChartData:

```
private string FormatChartData()
{
    SalesChartData chart = new SalesChartData();
    chart.data = new SalesSeriesPoint[12];

    chart.data[0] = new SalesSeriesPoint("Jan", CSSALES01);
    chart.data[1] = new SalesSeriesPoint("Feb", CSSALES02);
    chart.data[2] = new SalesSeriesPoint("Mar", CSSALES03);
    chart.data[3] = new SalesSeriesPoint("Apr", CSSALES04);
    chart.data[4] = new SalesSeriesPoint("May", CSSALES05);
    chart.data[5] = new SalesSeriesPoint("Jun", CSSALES06);
    chart.data[6] = new SalesSeriesPoint("Jul", CSSALES07);
    chart.data[7] = new SalesSeriesPoint("Aug", CSSALES08);
    chart.data[8] = new SalesSeriesPoint("Sep", CSSALES09);
    chart.data[9] = new SalesSeriesPoint("Oct", CSSALES10);
    chart.data[10] = new SalesSeriesPoint("Nov", CSSALES11);
    chart.data[11] = new SalesSeriesPoint("Dec", CSSALES12);

    JsonSerializerOptions serializerOptions = new JsonSerializerOptions();
    return JsonSerializer.Serialize<SalesChartData>(chart, serializerOptions);
}
```

Before adding the rest of implementation, let's stop here to explain.

FormatChartData starts by creating an instance of SalesChartData class (code will follow later). The object chart just created, has an uninitialized property called "data". This is a collection if SalesSeriesPoint objects (code will follow later). We allocate 12 members for this new collection.

Next, for every element in the collection, we create an object that represents the Sales series point, that is, the category name and its value. The value comes from the ***CSSALESnn*** property collection (which comes from the Workstation, table CUSREC DataSet).

Once the twelve month's data have been loaded, we use **.net** System.Text.Json classes to "serialize" our C# objects into a JSON object in string from. This is what the JavaScript expects to find as the HTML content for our Chart placeholder.

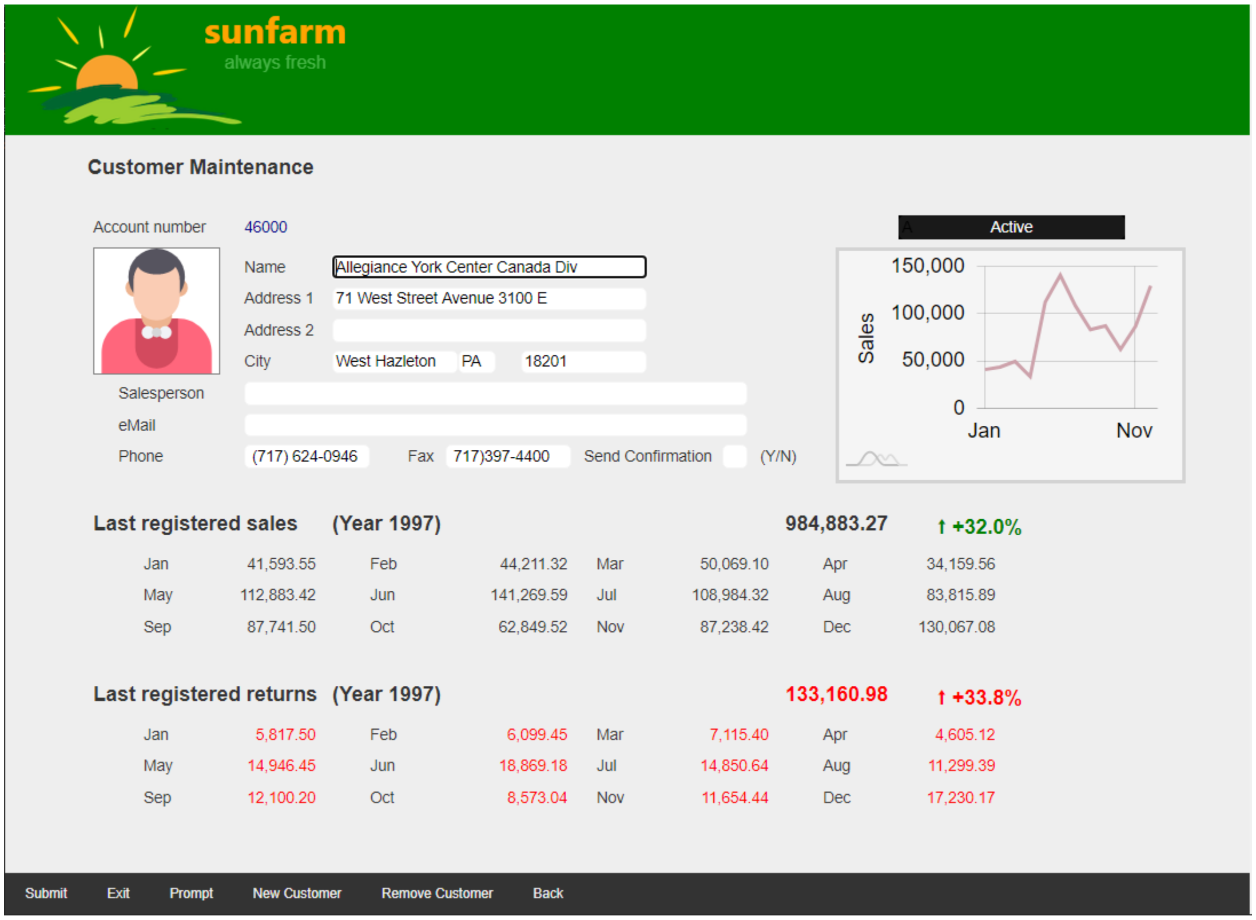
Since we are using Text.Json, let's add at the top of our C# source file the following line:

```
using System;
using System.ComponentModel.DataAnnotations;
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;
using ASNA.QSys.ExpoModel;
using System.Text.Json;
```

Then, let's implement the rest of the supporting small classes:

```
class SalesSeriesPoint
{
    public SalesSeriesPoint(string month, decimal sales)
      { this.month = month; this.sales = sales; }

    public string month { get; set; }
    public decimal sales { get; set; }
}

class SalesChartData
{
    public SalesSeriesPoint[] data { get; set; }
}
```

Run the Website Application with the latest changes, the following final Page should show: [5]



Customer Maintenance

Account number	46000
Name	Allegiance York Center Canada Div
Address 1	71 West Street Avenue 3100 E
Address 2	
City	West Hazleton PA 18201
Salesperson	
eMail	
Phone	(717) 624-0946
Fax	717)397-4400
Send Confirmation	(Y/N)

Last registered sales (Year 1997) 984,883.27 ↑ +32.0%

Jan	41,593.55	Feb	44,211.32	Mar	50,069.10	Apr	34,159.56
May	112,883.42	Jun	141,269.59	Jul	108,984.32	Aug	83,815.89
Sep	87,741.50	Oct	62,849.52	Nov	87,238.42	Dec	130,067.08

Last registered returns (Year 1997) 133,160.98 ↑ +33.8%

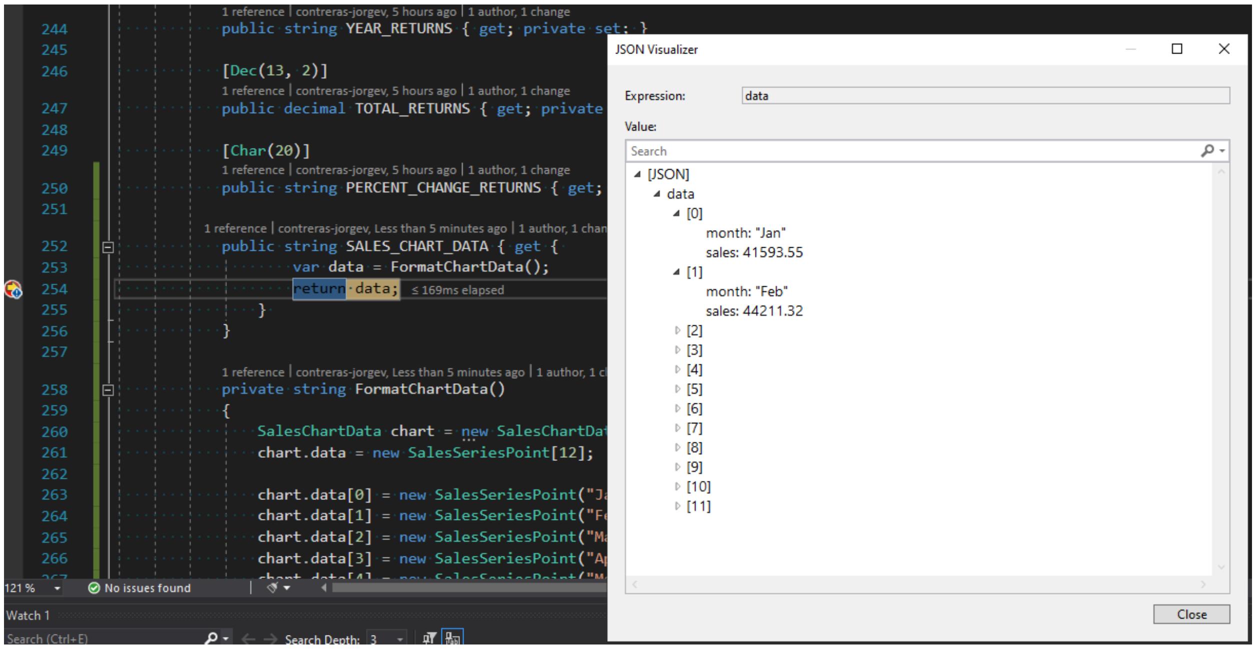
Jan	5,817.50	Feb	6,099.45	Mar	7,115.40	Apr	4,605.12
May	14,946.45	Jun	18,869.18	Jul	14,850.64	Aug	11,299.39
Sep	12,100.20	Oct	8,573.04	Nov	11,654.44	Dec	17,230.17

With the Chart you can quickly see that the Customer 46000 increased sales in 1997, and had a very good month in the middle of the year (June).

What a difference!

[5] Commit “Chart showing Sales data”

For the sake of completeness, you may want to break-up the property SALES_CHART_DATA, such that you can add a breakpoint and visualize the data points formatted in JSON format (you can look at the string representation as well as how the Visual Studio built-in JSON visualizer presents the data):



```
        1 reference | contreras-jorgev, 5 hours ago | 1 author, 1 change
        public string YEAR_RETURNS { get; private set; }

        [Dec(13, 2)]
        1 reference | contreras-jorgev, 5 hours ago | 1 author, 1 change
        public decimal TOTAL_RETURNS { get; private

        [Char(20)]
        1 reference | contreras-jorgev, 5 hours ago | 1 author, 1 change
        public string PERCENT_CHANGE_RETURNS { get;

      1 reference | contreras-jorgev, Less than 5 minutes ago | 1 author, 1 chan
        public string SALES_CHART_DATA { get {
                var data = FormatChartData();
                return data;  ≤ 169ms elapsed
            }
        }

        1 reference | contreras-jorgev, Less than 5 minutes ago | 1 author, 1 c
        private string FormatChartData()
        {
            SalesChartData chart = new SalesChartDat
            chart.data = new SalesSeriesPoint[12];

            chart.data[0] = new SalesSeriesPoint("Ja
            chart.data[1] = new SalesSeriesPoint("Fe
            chart.data[2] = new SalesSeriesPoint("Ma
            chart.data[3] = new SalesSeriesPoint("Ap
```

JSON Visualizer

Expression: data

Value:

```
◢ [JSON]
  ◢ data
    ◢ [0]
        month: "Jan"
        sales: 41593.55
    ◢ [1]
        month: "Feb"
        sales: 44211.32
    ▷ [2]
    ▷ [3]
    ▷ [4]
    ▷ [5]
    ▷ [6]
    ▷ [7]
    ▷ [8]
    ▷ [9]
    ▷ [10]
    ▷ [11]
```