Enhancing Expo Displayfile Guide

Overview

This guide will show basic techniques to change the look and feel Monarch Migrated application Pages, to make them more appealing.

During the description of the steps to apply the different techniques, the public repository in GitHub will be used

---

Reference public GitHub repository:

https://github.com/ASNA/SunFarm

The repository was created with a fresh new Monarch Migration. As enhancements were being applied, the required changes were committed, such that by using git History tooling, the changes may be seen in real code.

Assumptions

Familiarity with the ASNA ***Customer Sample IBMi program*** is required. This program has been provided by ASNA as sample RPG/CL code in several products, with the names: M4CUST, M5CUST and/or IronMonger.

Scope

This Guide shows how to Enhance the following Pages:
1. Customer Inquiry (Initial Page)
2. Customer Maintenance (Option “2”)

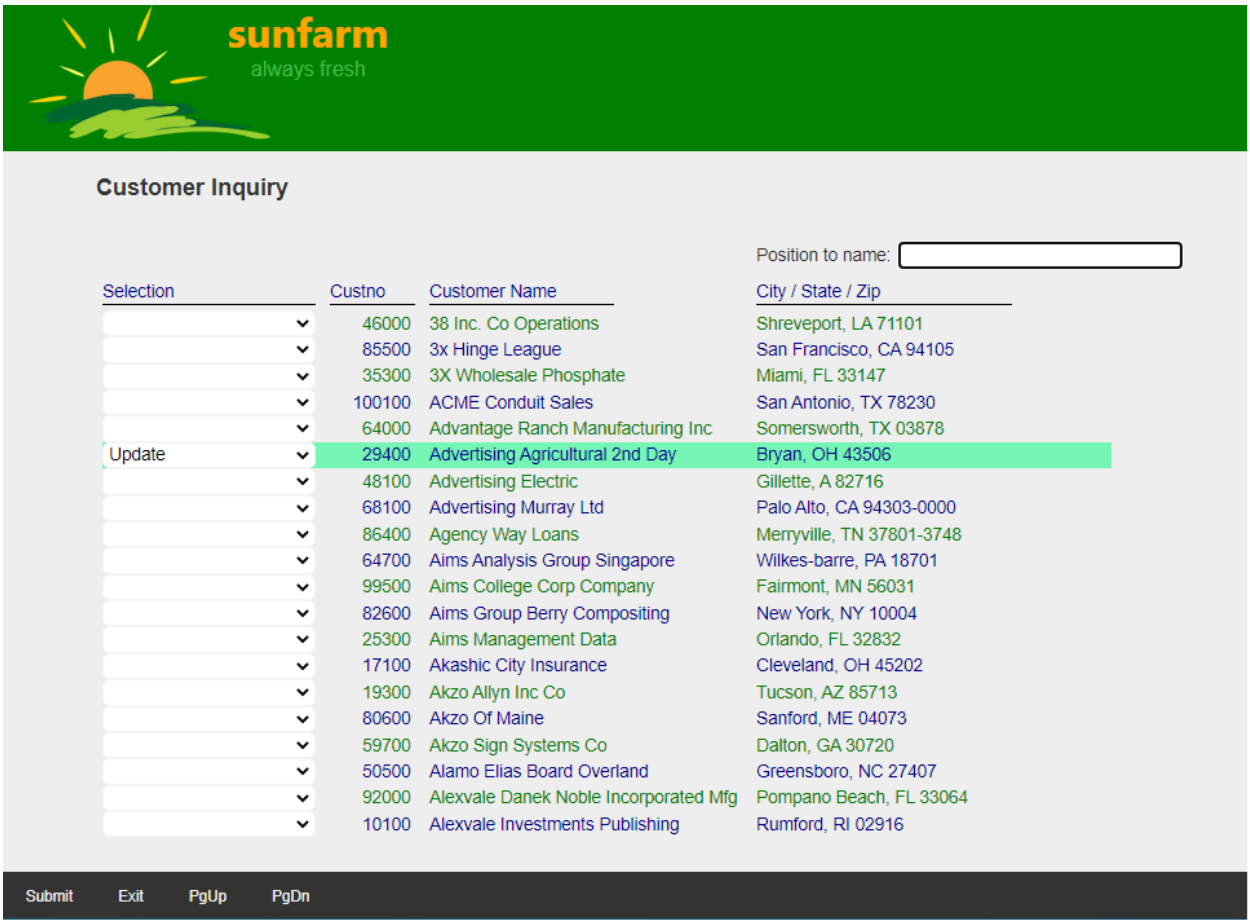
In addition, the Site has been branded to show the fictitious company “SunFarm” with its logo on all pages.

What this Guide *is not*

1. This Guide is not an introduction to ASNA Core Foundation
2. This is not a Tutorial.
3. Cloning the repository is not enough. There are ASNA references (or source Projects) that need to be installed.

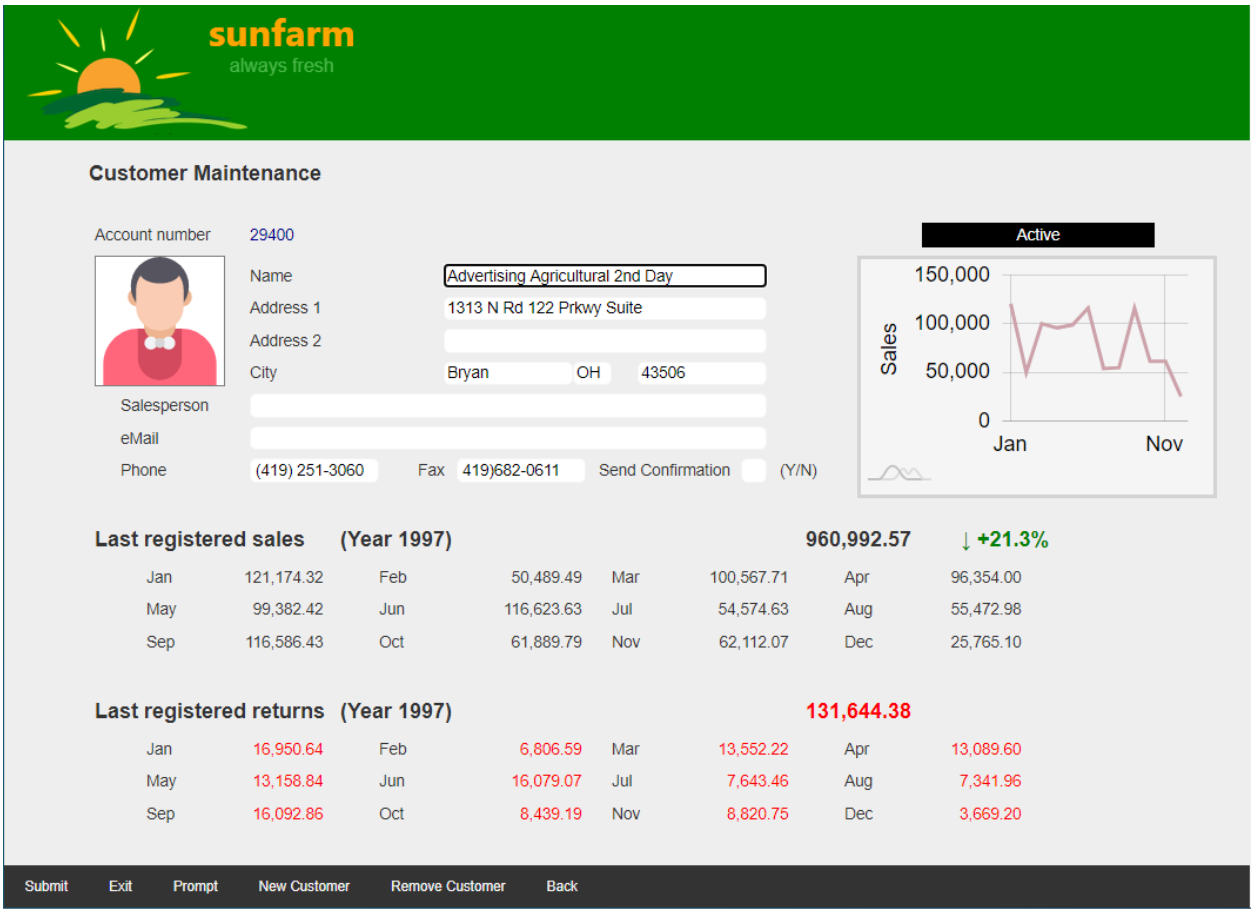
Specifically, the goal is to transform: Customer Inquiry Page:

To this modern look:



Customer Maintenance Page:

To the page where more useful information is displayed



Data Files used

The IBMi data files were also migrated to a Microsoft SQL instance.

Microsoft SQL Server Management Studio has been used to show file definitions and query results.

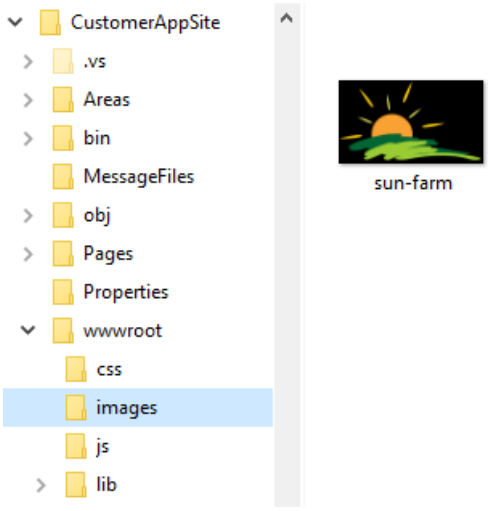
The DataGate database in $\SunFarm\CustomerAppLogic\MyJob.cs named ***NancySQL*** points to a Microsoft SQL instance where the Data Files have been migrated.

Adding branding: SunFarm Logo as a heading on all pages.

The heading is produced by and image, text added with proper CSS and a solid background.

The file used is a PNG raster image as shown, with transparency. The name of the file is “sun-farm.png”

The place where static resources are placed is under the `wwwroot` folder:



Out of the box, the folder ***wwwroot/css*** already has a CSS file named ***site.css***.
It is empty.

Let’s add some styles that will allow us to position and define the look of our logo elements

```
#logo-banner {
    height: auto;
    background-color: green;
}

#logo {
    padding-left: 1em;
}

#logo-text {
    display: inline-block;
    vertical-align: top;
    margin-left: -41px;
}

#logo-title {
    color: orange;
    font-size: xx-large;
    font-family: system-ui;
    font-weight: bolder;
}

#logo-subtitle {
    text-align: center;
    color: lightgreen;
    font-size: medium;
    opacity: 0.5;
}
```
[1]

The markup that we need is standard HTML which we will place in the file:

`$SunFarm\CustomerAppSite\Areas\CustomerAppViews\Pages\_ViewStart.cshtml.cshtml`

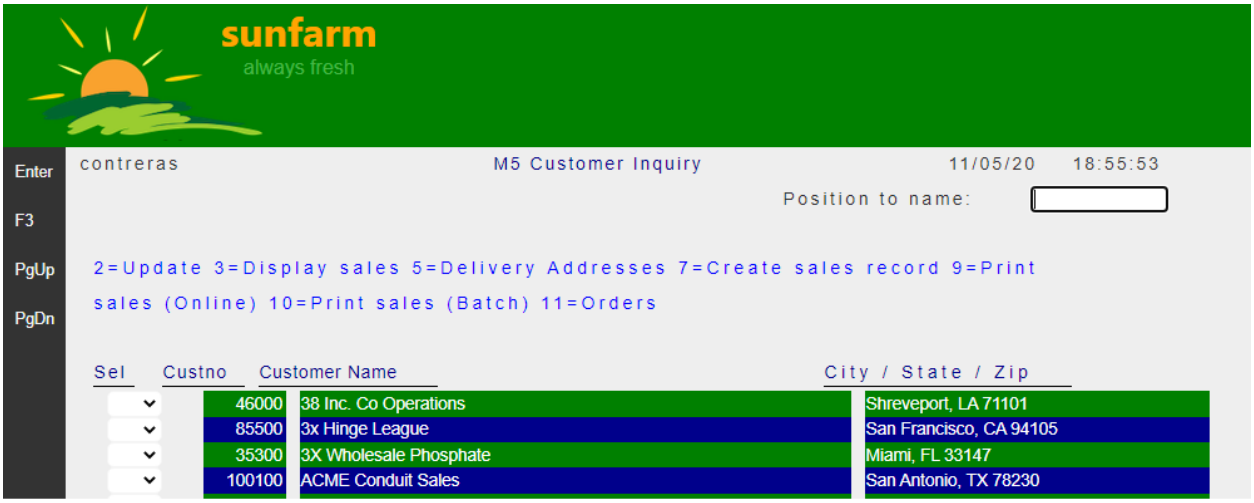
This file shared with all the Pages in the View file structure.

```
@{
    Layout = "_Layout";
}

<div id="logo-banner">
    <img id="logo" src="~/images/sun-farm.png" />
    <div id="logo-text">
        <div id="logo-title">sunfarm</div>
        <div id="logo-subtitle">always fresh</div>
    </div>
</div>
```

The Application will show the branded heading on top of all pages, as shown below:

---

[1] Commit: “Logo resources”.



2

Navigation menu (formerly DDS File Active Function Keys)

The two pages we are enhancing, are formed by records described by `CUSTDSP` Displayfile.

All Display files are described using Razor Page syntax, where ExpoTags runtime support augments with DDS-like tag-helpers.

The Expo tag-helper that represents the *“DDS File Active Function Keys”* is named ***DdsFunctionKeys*** (tag-helper).

The default markup for Monarch Migrated Displayfiles shows the ***DdsFunctionKey*** tag-helper without any attributes (at the top of each Displayfile, inside the ***DdsFile*** tag-helper):

```
<form id="MonarchForm" method="post">
    <DdsFile DisplayPageModel="Model">

        <DdsFunctionKeys />

        <main role="main" class="display-element-uninitialized">
            .
            .
            .
```

[2] Commit: Branding all pages.

Let's change the ***Location*** attribute to the value "HorizontalBottom":

```
<DdsFunctionKeys Location="HorizontalBottom" />
```

The Navigation Menu looks much better at the bottom of the Page:

Instead of using aid key, you may want to use the mover vernacular form function key, something like: "Exit" equivalent to the **F3** aid key being pressed at the keyboard.

Note: experienced users may still use the ***Fxx*** key, if there is a physical keyboard that produces such key-code.

3

[3] Commit: Navigation menu customization

Re-labeling the Menu Options

Expo Tag-helpers that represent options at the DDS File (*DdsFile*) level and DDS Records (*DdsRecord* & *DdsSubfileControl*) have, have an attribute called ***KeyNames***, where we can override the default active key labels.

```
KeyNames="ENTER 'Submit'; F3 'Exit'; PageUp '◄ Page'; PageDown 'Next ►';"
```

Where the value in single-quotes re-labels the text for the key name that precedes it. Instead of "ENTER" we show 'Submit'; instead of "F3" we show 'Exit', etc.

Note: The labels for PgUp and PgDn should have been replaced. This bug has been reported to R&D.

Color effects that worked on *green-screen* are displeasing on a modern *blue-screen* Browser Pages

DDS on the IBMi usually renders on a terminal with dark background. Text screen attributes such as REVERSE-IMAGE produced a subtle lighter green background which some considered pleasant on the eye to highlight text, such as record on a subfile.

REVERSE-IMAGE on a typically white/pale background Browser Page would produce a very heavy block on text when rendered.

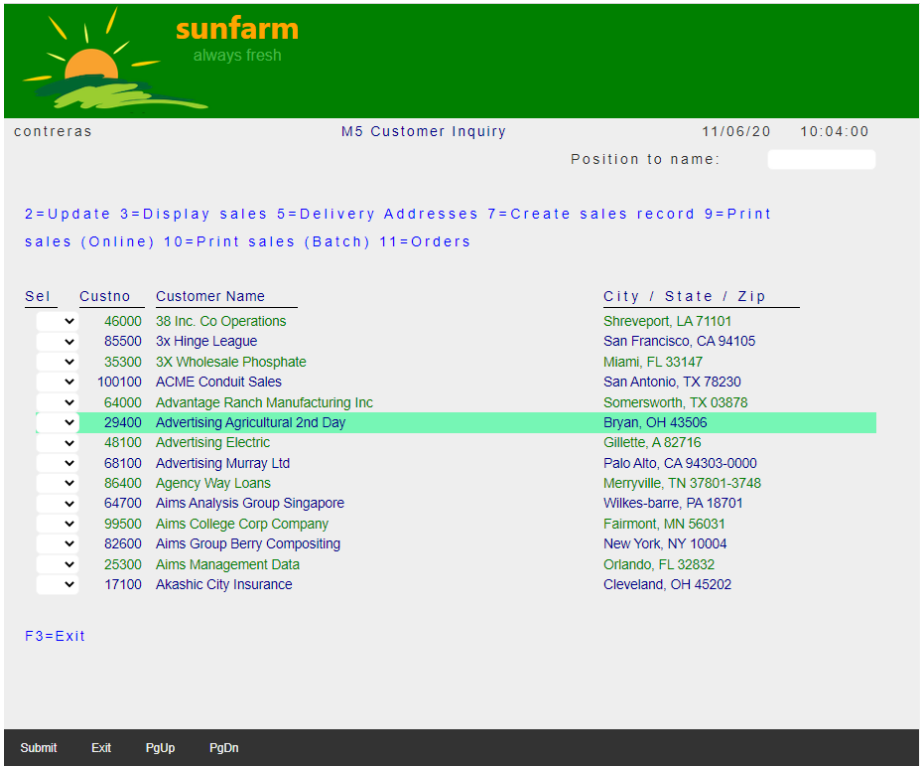
As you can see on the off-the-shelf migration for "M5 Customer Inquiry" Page, the subfile records are rendered with a color that is displeasing to the eye.

The reverse-image display attribute DDS keyword **DSPATR(RI)** gets translated to **InvertFontColors="*True"** tag-helper attribute.

Removing such attribute produces a much pleasant output.

$\SunFarm\CustomerAppSite\Areas\CustomerAppViews\Pages\CUSTDSPF.cshtml
Lines 51, 52 and 53:[4]

```
<div Row="8" RowSpan="@SFLC_SubfilePage * @SFLC_SubfileRowsPerRecord">
    @for (int rrn=0, row = 8; rrn < Model.SFLC.SFL1.Count; rrn++, row +=
@SFLC_SubfileRowsPerRecord)
    {
        <DdsSubfileRecord RecordNumber="rrn" For="SFLC.SFL1">
            <div IsGridRow>
                <DdsCharField Col="2" For="SFLC.SFL1[rrn].SFCOLOR" VisibleCondition="*False"
VirtualRowCol="@row,2" tabIndex=1 />
                <DdsDecField Col="4" For="SFLC.SFL1[rrn].SFSEL" VirtualRowCol="@row,4"
EditCode="Z" ValuesText="'0','2','3','5','7','9','10','11'" tabIndex=2 />
                <DdsDecField Col="7+1" For="SFLC.SFL1[rrn].SFCUSTNO" VirtualRowCol="@row,7"
Color="Green : !61 , DarkBlue : 61" InvertFontColors="*True" EditCode="Z" Comment="CUSTOMER
NUMBER" />
                <DdsCharField Col="14+1" For="SFLC.SFL1[rrn].SFNAME1" VirtualRowCol="@row,14"
Color="Green : !61 , DarkBlue : 61" InvertFontColors="*True" />
                <DdsCharField Col="55+1" For="SFLC.SFL1[rrn].SFCSZ" VirtualRowCol="@row,55"
Color="Green : !61 , DarkBlue : 61" InvertFontColors="*True" Comment="CITY-STATE-ZIP" />
            </div>
        </DdsSubfileRecord>
    }
</div>
```



[4] Commit: Remove Reverse-image on Subfile records

Removing Redundant green-screen typical Information

Terminal program Displays usually included information about system Date and Time. Often the username was also included in most pages.

Modern devices that provide Web Browsers have already a place for this information to be retrieved.

Other green-screen constants that provided assistance to the user, such as which aid keys were active, are now part of the “tooltip” information that is already part of the Navigation Menu.

Lastly, constants that listed available options, in this case for selection of records in the subfile, can be placed in a better place.

The next commit will:
1. Remove User, Date and Time fields.
2. Remove F3=Exit constant.
3. Move the constant “2=Update 3=Display sales …” to the “SFLC.SFL1[rrn].SFSEL" field as Value-Text (or pull down selection option labels)

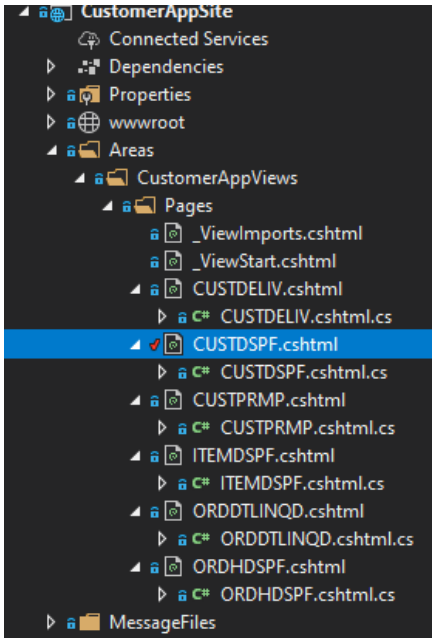
```
$\SunFarm\CustomerAppSite\Areas\CustomerAppViews\Pages\CUSTDSPF.cshtml

<div Row="1">
    <DdsConstant Col="2" Text="*USER" />
    <DdsConstant Col="31+1" Text="M5 Customer Inquiry" Color="DarkBlue" />
    <DdsConstant Col="64+1" Text="*DATE" />
    <DdsConstant Col="73+1" Text="*TIME" />
</div>


<div Row="4">
    <DdsConstant Col="3" Text="2=Update  3=Display sales  5=Delivery Addresses  7=Create sales record  9=Print" Color="Blue" />
</div>
<div Row="5">
    <DdsConstant Col="3" Text="sales (Online)  10=Print sales (Batch)  11=Orders" Color="Blue" />
</div>

<DdsRecord For="KEYS" KeyNames="ENTER 'Submit'; ">
    <div Row="23">
        <DdsConstant Col="3" Text="F3=Exit" Color="Blue" />
    </div>
</DdsRecord>
```

The Subfile field for the options, deserves further explanation.

We have two pieces of information:
1. The option code (char value), that is: <nothing>, 2, 5, 7, 9, 10 and 11.
2. The corresponding labels, such as: “Update” “Display sales” “Delivery Addresses” etc.

Each ASP Razor Page is defined by two files: the Markup file (extension *.cshtml*) and the corresponding Model file (extension *.cshtml.cs*).

For convenience, Visual Studio Solution Explorer, shows the Model file under the Markup file in the Website file structure:



Visual Studio intellisense, allows to jump back and forth, between symbols defined in the Markup and the Model. For example, positioning the cursor in the markup on top of DdsDecField For="SFLC.SFL1[rrn].SFSEL" and pressing F12, will take you to the Model's definition for the SFLC (Subfile record Controller) 's field SFLSEL (in blue):

```
public class SFL1_Model : SubfileRecordModel
{
    [Char(1, Protect = "*True")]
    public string SFCOLOR { get; set; }

    [Values(typeof(Decimal),"00","02","03","05","07","09","10","11")]
    [Dec(2, 0)]
    public decimal SFSEL { get; set; }

    [Dec(6, 0)]
    public decimal SFCUSTNO { get; private set; } // CUSTOMER NUMBER

    [Char(40)]
    public string SFNAME1 { get; private set; }

    [Char(25)]
    public string SFCSZ { get; private set; } // CITY-STATE-ZIP
}
```

Note that, in addition to C# decimal type, the field SFLSEL is decorated with Dec and Values attributes.

Dec attribute further defines the decimal as one with fixed precision and decimal positions. *Values* attributes define the valid values for the field. The position of the list of valid values is important, since it will be matched with the *ValuesText* tag-helper attribute.

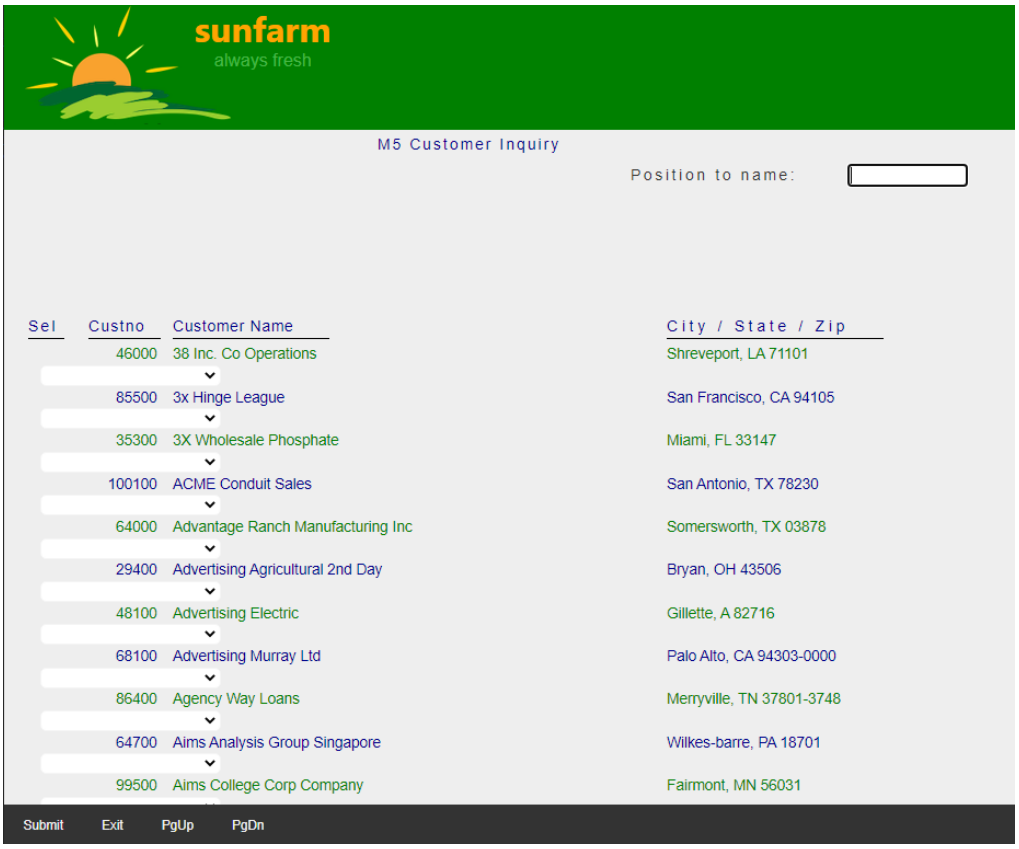
Back on the Markup file: …\Areas\CustomerAppViews\Pages\CUSTDSPF.cshtml

```
<DdsDecField Col="4"
  For="SFLC.SFL1[rrn].SFSEL"
  VirtualRowCol="@row,4"
  EditCode="Z"
  ValuesText="'0','2','3','5','7','9','10','11'"
  tabIndex=2
/>
```

Let's change ValuesText as follows:

```
<DdsDecField Col="4"
  For="SFLC.SFL1[rrn].SFSEL"
  VirtualRowCol="@row,4"
  EditCode="Z"
  ValuesText="' ','Update','Display sales','Delivery Addresses','Create sales
record','Printsales (Online)','Print sales (Batch)','Orders'"
  tabIndex=2
/>
```

[5]



Now we need to push the rest of the fields to the right, to make it look nicer.

There are several ways to accomplish this. What we will use, is the trial-and-error technique.

---

[5] Subfile selection options as pull-down options.

To avoid having to recalculate the column positions for the fields in the subfile, we can add a constant numeric value to each of the fields to the right: “SFLC.SFL1[rrn].SFCUSTNO”, “SFLC.SFL1[rrn].SFNAME1" and “SFLC.SFL1[rrn].SFCSZ”.

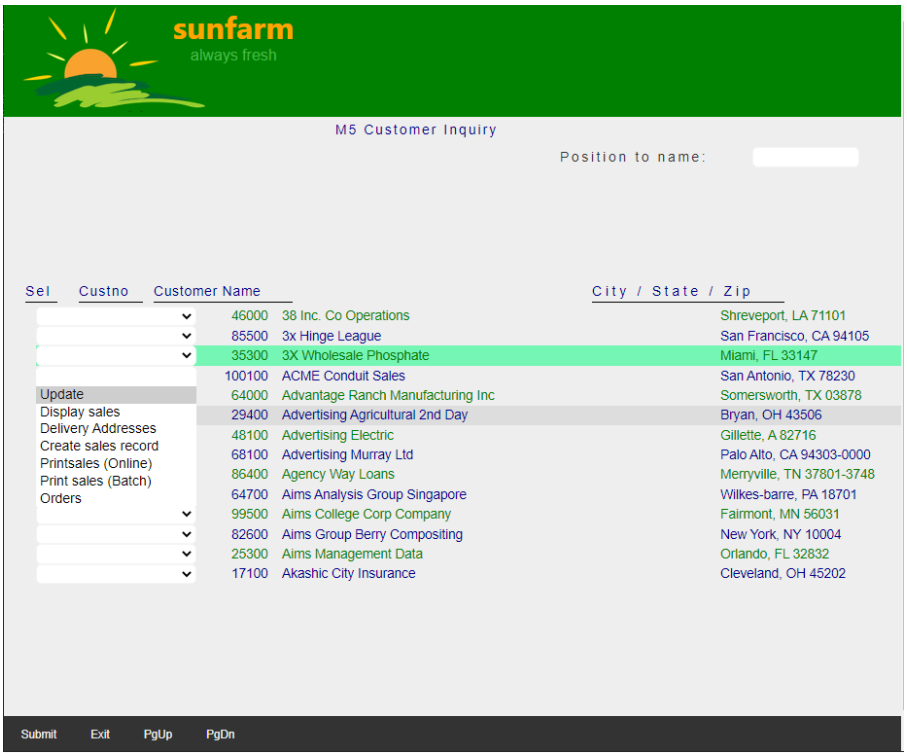
Let’s add *10*+ … no, *15*+ … no *12*+

The markup would look like this (for clarity I eliminated part of each line, replaced by …):

```
<DdsSubfileRecord RecordNumber="rrn" For="SFLC.SFL1">
    <div IsGridRow>
        <DdsCharField Col="2" For="SFLC.SFL1[rrn].SFCOLOR" ...
        <DdsDecField Col="4" For="SFLC.SFL1[rrn].SFSEL" ...
        <DdsDecField Col="12+7+1" For="SFLC.SFL1[rrn].SFCUSTNO" ... Comment="CUSTOMER NUMBER" />
        <DdsCharField Col="12+14+1" For="SFLC.SFL1[rrn].SFNAME1" ...  />
        <DdsCharField Col="12+55+1" For="SFLC.SFL1[rrn].SFCSZ" ... Comment="CITY-STATE-ZIP" />
    </div>
</DdsSubfileRecord>
```

As you change the markup, you may refresh the running page in the Browser, until it looks how you want it.

Note that the Col attribute tag-helper, already came with an expression. Cocoon Displayfile Migration Agent took advantage of the expression to indicate the ***original*** column value (from DDS and adjustments it had to do to prevent overlaps — due to HTML borders and padding—



Before we continue, let’s get rid of the vertical gap, between Position to name: (Row=2) and the headings of the Subfile (Row=7).

We accomplish this by changing ~~Row=“7”~~ to Row=“3” and ~~Row=“8”~~ to Row=“4”

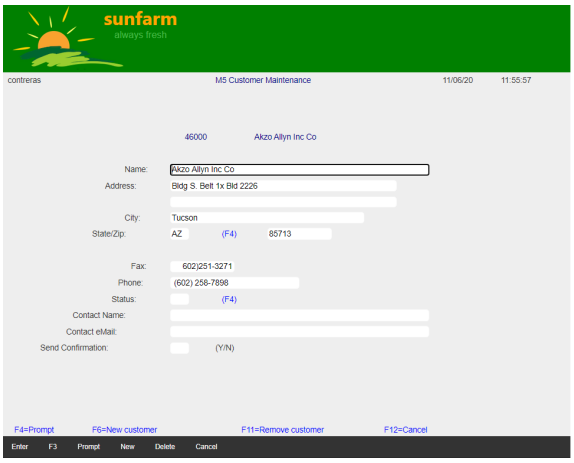
6

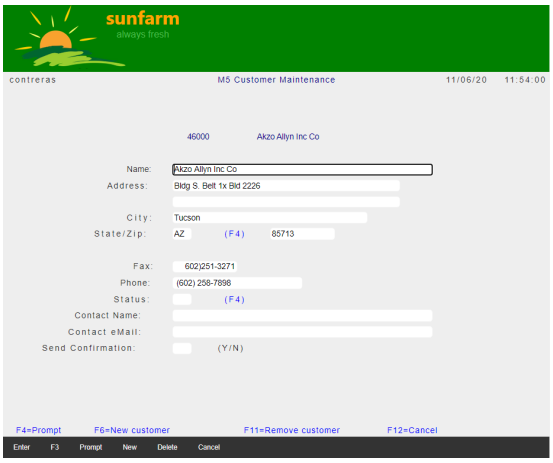
Stretching constant label's Text

If you had not noticed, the text on the constants is rendered with more white space between characters. Particularly noticeable is the subfile column heading "City / State / Zip".

ASNA Expo DdsConstant tag-helper has an attribute called ***StretchConstantText*** which defaults to ***true***. That attribute can be overridden at the Record level.

That attribute exists to match better the constant text alignment that green-screen developers used, particularly when splitting words in multiple DDS constants, or when right justifying (manually) text on the screen.





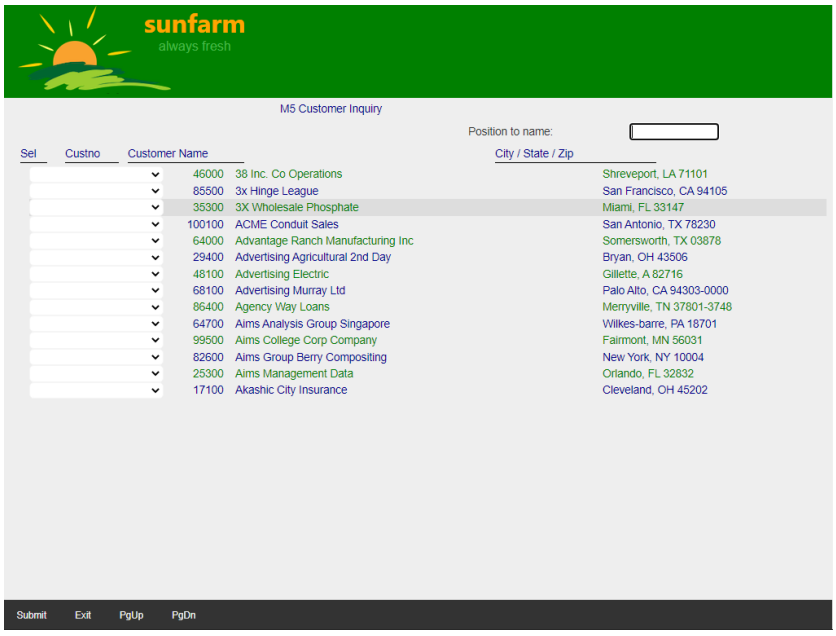
The left image shows Stretching Off and the right Stretching On. Notice how the right page improves the label right-alignment. *With further CSS adjustments it can be made even closer to what the green-screen designer intended* — out of the scope of this Guide—

[6] Commit: Push subfile up.

For the Customer Inquiry (first page) we will reset the constant stretching effect, by adding StretchConstantText attribute:

```
<DdsSubfileControl For="SFLC"
  StretchConstantText=false
  KeyNames="ENTER 'Submit'; F3 'Exit'; PageUp '◄ Page'; PageDown 'Next ►';"
  SubfilePage="@SFLC_SubfilePage"
  CueCurrentRecord=true ClickSetsCurrentRecord=true
>
```

Almost complete:



Page Title with standard HTML/CSS

We can add HTML/CSS when Grid alignment serves no purpose. For example, we want the Title of the page to show aligned to the Page's left and using standard CSS styles.

Add the following CSS style to file

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

```
#page-title {
    font-size: large;
    padding-left: 4.0em;
    padding-top: 1em;
    font-weight: bold;
}
```

And replace the following row / col constant definition in the CUSTDSP.chtml markup:

```
    <div Row="1">
        <DdsConstant Col="31+1" Text="M5 Customer Inquiry" Color="DarkBlue" />
    </div>
```

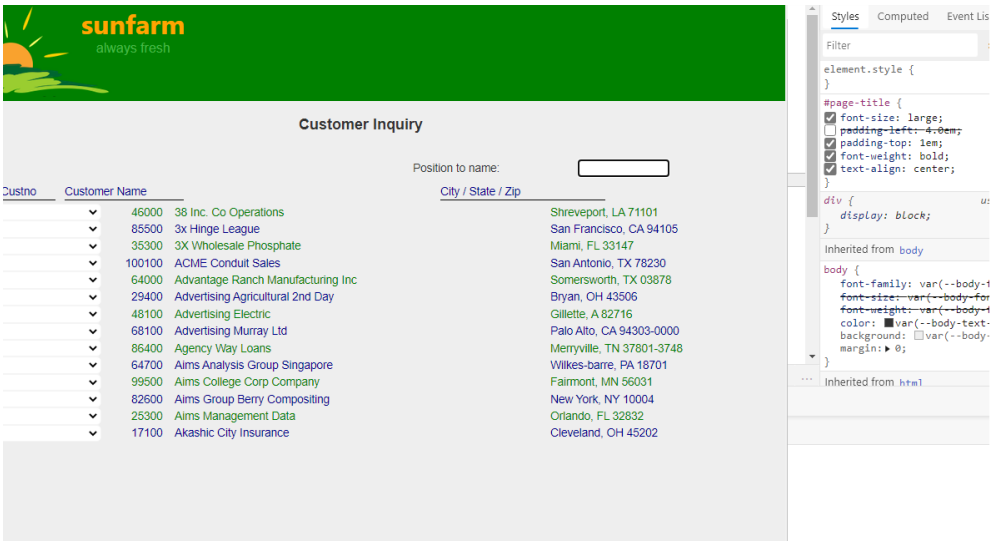
With this standard markup:

```
  <div id="page-title">Customer Inquiry</div>
```

[7]

This may be particularly important when centering titles on a Page.

You may even user the Browser's developer tools to experiment with different styles:



[7] Commit: Replacing Page Title

Grid Column Span Adjustment

Field's starting positions are very accurately identified on the page based on the original DDS row, col positions. But ending column positions are harder.

The default Font for Expo Displayfiles is of type ***variable-pitch***, meaning that the width of characters varies according to the Font's designer's stroke used. Typically the letter "i" uses a lot less character width that an upper case "M".

Green-screen page designers used a Font that is of type fixed-pitch, meaning that the width of ALL characters is the same.

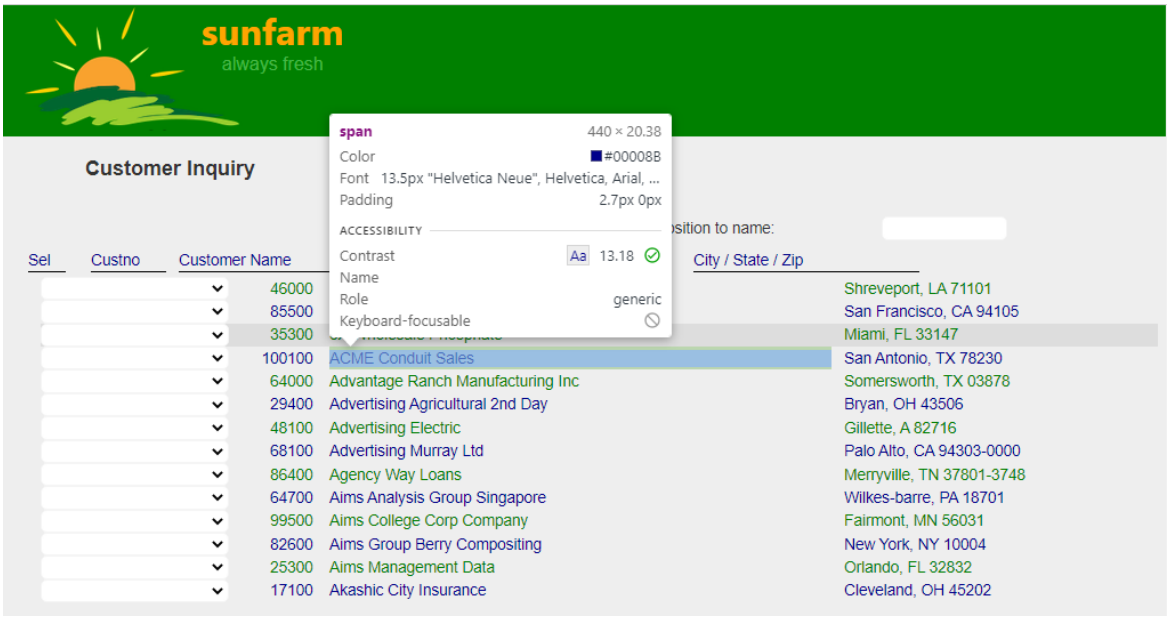
A green-screen label starting at column 5 with the constant "THIS CONSTANT" (thirteen characters) is guaranteed to end at column position 17. That is, the end-column position can be precisely computed by starting-position + field-length.

This no longer works on Browser fonts (even with those so-called *Monospaced*).

Monarch Cocoon Displayfile for Core Agent will use the length of the field of constant to compete the Grid column span (ending position), but with a fudge-factor to account for the use of Web fonts.

We can adjust that calculation on the field level.

Consider the column for SFNAME1 Subfile on the Customer Inquiry Page:

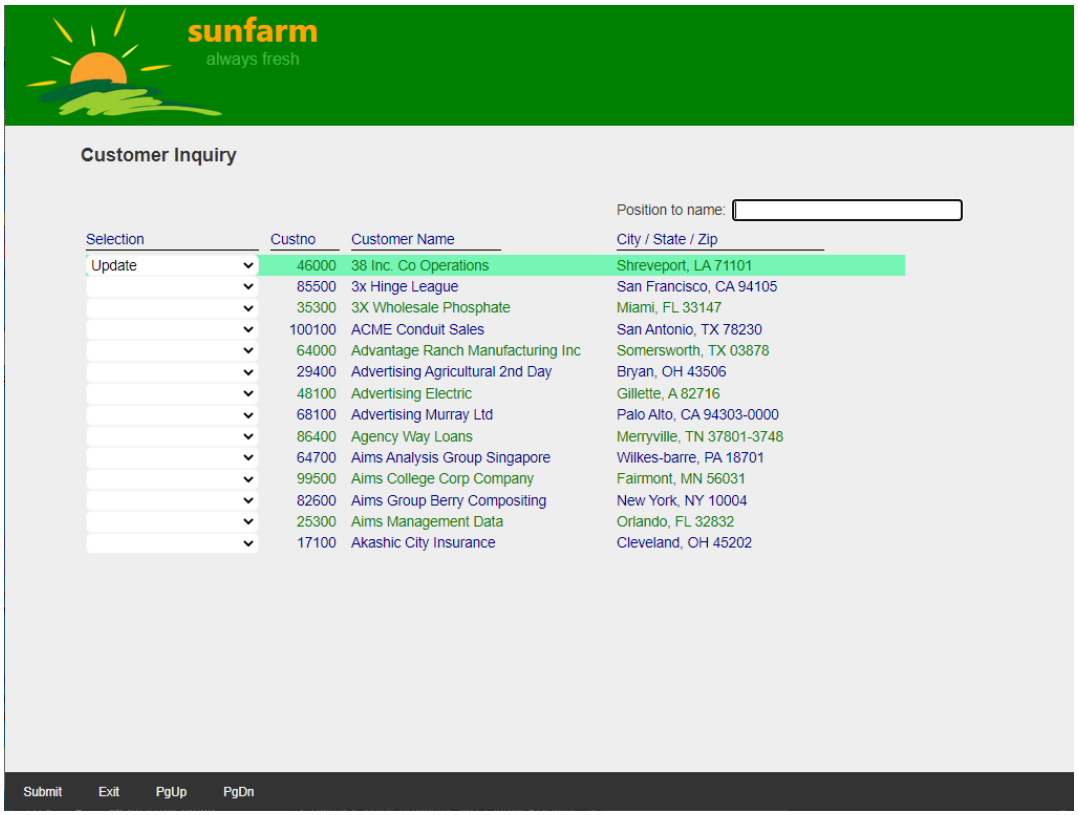


Notice how the field is defined as 40 characters, but most of the record's field value will likely not use the full 40.

We can override the column-span calculation by adding the tag-helper attribute ColSpan:

```
<DdsCharField Col="12+14+1" ColSpan="30" For="SFLC.SFL1[rrn].SFNAME1" VirtualRowCol="@row,14"
Color="Green : !61 , DarkBlue : 61" />
```

This way we can reduce Grid column positions and complete adjusting our elements on the Page to achieve the following look:
[8]



When Changes on Razor Pages is not enough

There comes a time when to continue enhancing pages, it is necessary to change the Business Logic.

Since we have been able to compress the screen while aligning elements and eliminating redundant items, we are left with unused real-estate.

The Business Logic is writing fourteen records at a time, now we can fit at least twenty (a nice rounding number) or even more if the majority of users may have higher resolution devices.

Let's assume we want to increase the record count from fourteen to twenty.

---

[8] Commit: Fourteen records subfile.

Displaying Twenty records per page in the Subfile

Two changes are needed:
1. Write twenty records to the subfile in the Business Rules.
2. Expand the record count to show in the Razor Page's Subfile Controller

We will use blue color for the code that needs to be added. Red strikeout for code that needs to be removed.

To change the Business Logic, we need to change the following class (program):

`$\SunFarm\CustomerAppLogic\CUSTINQ.cs`

```
public partial class Custinq : ASNA.QSys.HostServices.Program
{
    const int SFLC_SubfilePage = 20;

    protected Indicator _INLR;
.
.
.
    void LoadSfl() // Line 519
    {
        _IN[61] = '0'; //Start with green.
        _IN[90] = '1'; //Clear the subfile.
        CUSTDSPF.Write("SFLC", _IN.Array);
        _IN[76] = '0'; //Display records.
        _IN[90] = '0';
        sflrrn = 0;
        _IN[77] = CUSTOMERL2.ReadNext(true) ? '0' : '1';
        //-------------------------------------------------------------
        while (!(bool)_IN[77] && (sflrrn < 14 SFLC_SubfilePage))
        {
.
.
.
        //*********************************************************************
        //  Read Backwards for a PageDown
        //*********************************************************************
        void ReadBack() // Line 558
        {
            _IN[76] = '0';
            _IN[77] = '0';
            X = 0;
            CUSTDSPF.ChainByRRN("SFL1", 1, _IN.Array); //Get the top name and
            CMNAME = SFNAME1; // number.
            CMCUSTNO = (decimal)SFCUSTNO;
            CUSTOMERL2.Chain(true, CMNAME, CMCUSTNO);
            _IN[76] = CUSTOMERL2.ReadPrevious(true) ? '0' : '1';
            while (!(bool)_IN[76] && (X < 14 SFLC_SubfilePage))
            {
                /* EOF or full s/f. */
                X += 1;
                _IN[76] = CUSTOMERL2.ReadPrevious(true) ? '0' : '1';
            }
            if ((bool)_IN[76])
                //Any records found?
                CUSTOMERL2.Seek(SeekMode.SetLL, new string(char.MinValue, 40));
        }
```

There are two places in CUSTINQ where the hard-coded value 14 is used, representing the records to write to the subfile.

We define a constant, and use it instead so we can adjust that number to twenty.
To expand the number of records that need to be displayed in the Subfile controller, we need to affect two files:

1. Model File: $\SunFarm\CustomerAppSite\Areas\CustomerAppViews\Pages\CUSTDSPF.cshtml.cs
2. Markup File: $\SunFarm\CustomerAppSite\Areas\CustomerAppViews\Pages\CUSTDSPF.cshtml

In the Model file, we need to update the Subfile Controller's ***Size*** attribute:

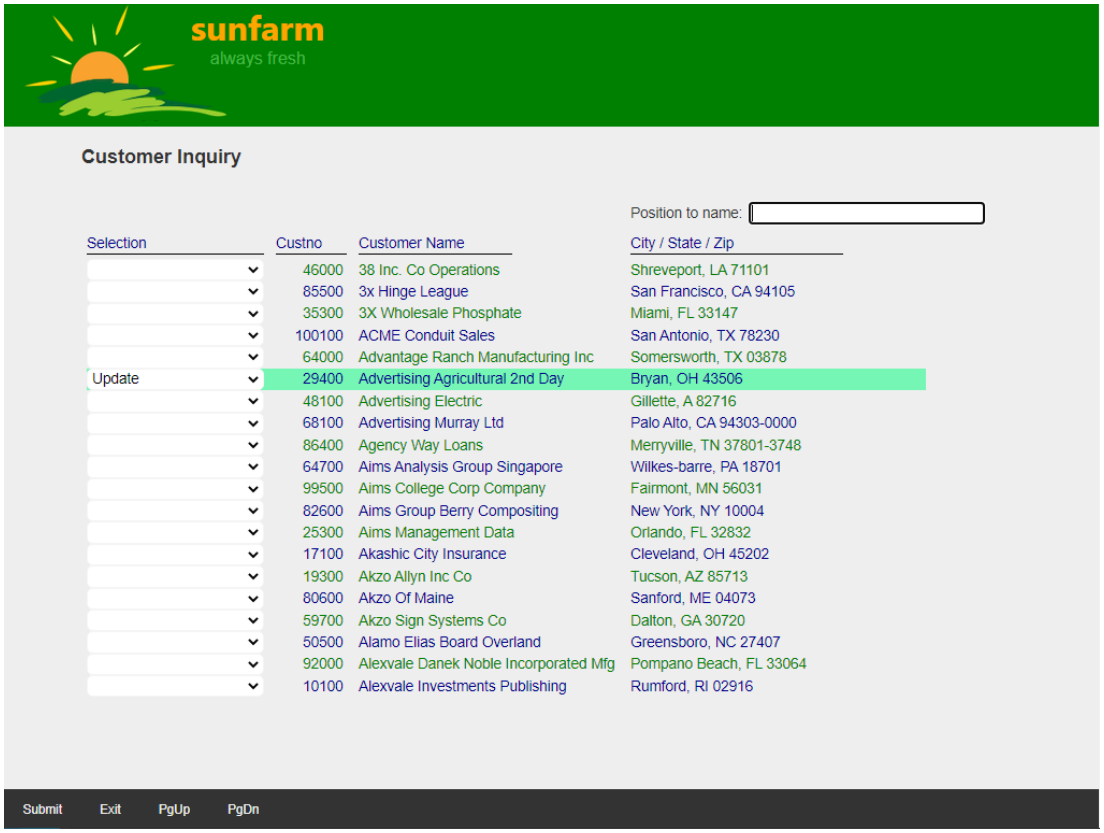
```
[
    SubfileControl(ClearRecords : "90",
        FunctionKeys = "PageUp 51:!76;PageDown 50:!77",
        DisplayFields = "!90",
        DisplayRecords = "!90",
        Size = ~~14~~ 20,
        IsExpandable = false,
        EraseFormats = "CUSTREC SALESREC"
    )
]
public class SFLC_Model : SubfileControlModel
```

In the Markup file, we just need to change the value of the variable that controls the page size:

```
  @{
      int SFLC_SubfilePage = ~~14~~ 20;
      int SFLC_SubfileRowsPerRecord = 1;
  }
  <DdsSubfileControl For="SFLC" StretchConstantText=false KeyNames ...
```

Build the Business Logic Project (CustomerAppLogic) …
Build the Website (CustomerAppSite) …

If all goes well, twenty records per page should show in the Subfile:
[9]



[9] Commit: Customer Inquiry Twenty records per page.