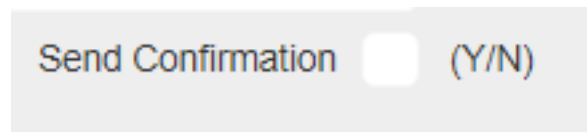


Replacing YES/NO fields with Check-boxes

Green-screen pages typically used boolean fields with a one-char field that expects the value 'Y' or 'N'.

In the second redesigned page, we have the field called CUSTREC.SFYNO1 it is rendered as:

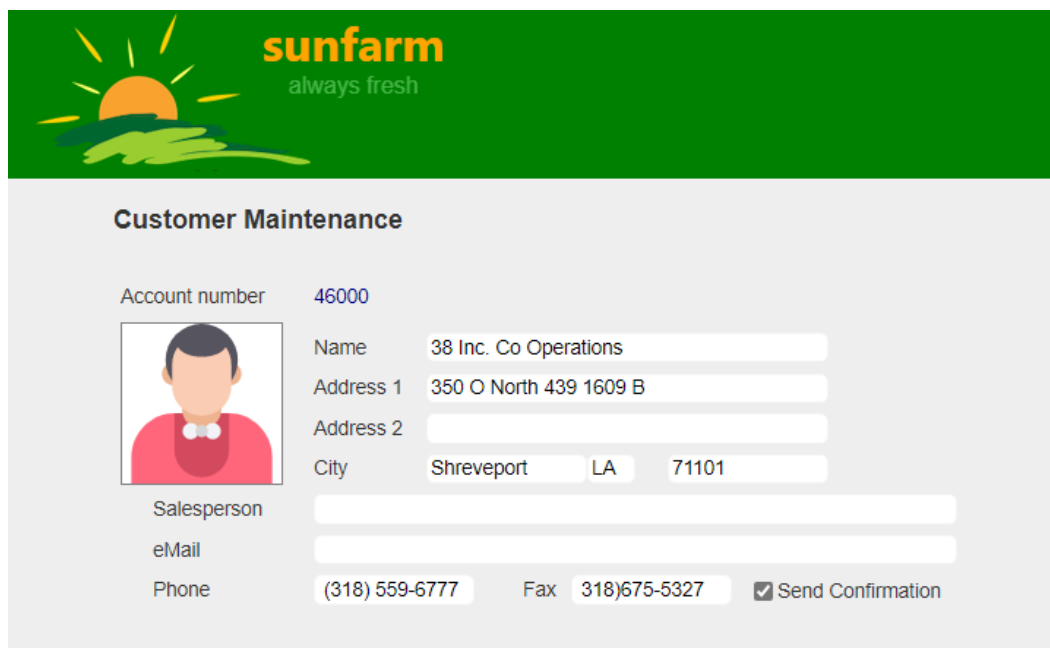
A light gray rectangular box containing the text 'Send Confirmation' in a blue, monospace-style font, followed by a small white square input field, and then the text '(Y/N)' in a red, monospace-style font.

Using character fields for booleans presents several problems:

1. They need a constant to the right to show the possible expected values, in this example **(Y/N)**. Not only does it waste valuable screen real-estate, but also requires the constant to be properly aligned.
2. Having a character field that allows more than two possible input values, makes the interface error prone and therefore more validation will be required.

Checkboxes on Web pages are usually presented with the “tick” button to the left of the label, with a specific spacing between the button and the label. It should make sense then to include the constant label, such as “Send Confirmation” as part of the Tag helper.

What we want to produce is the following:

A modern web form titled 'Customer Maintenance' with a green header featuring a sun logo and the text 'sunfarm always fresh'. The form contains several input fields: 'Account number' (46000), a profile picture placeholder, 'Name' (38 Inc. Co Operations), 'Address 1' (350 O North 439 1609 B), 'Address 2', 'City' (Shreveport, LA, 71101), 'Salesperson', 'eMail', 'Phone' ((318) 559-6777), 'Fax' (318)675-5327, and a checked checkbox for 'Send Confirmation'.

When the “tick” is checked, the application logic wants the field with a character value of “Y” and when it does not a “N” (or a blank)

The Markup can be simplified too, from three lines:

```
<DdsConstant Col="47" Text="Send Confirmation" />
<DdsCharField Col="58" ColSpan="2" For="CUSTREC.SFYN01" VirtualRowCol="18,27" />
<DdsConstant Col="61" Text="(Y/N)" />
```

Down to one:

```
<DdsCheckboxField Col="47" Text="Send Confirmation" For="CUSTREC.SFYN01" VirtualRowCol="18,27" />
```

On the Model source file, we decorate the field, providing information for the value we want to mean **checked** (or true), and which for **unchecked** (or false).

The Values attribute for **DdsCheckboxField** expects the first value to be the checked and the second the unchecked.

1

```
[Char(1)]
[Values(typeof(string), "Y", "N")]
public string SFYN01 { get; set; }
```

Note: DdsCheckboxField can be used on Decimal workstation fields too.

Using Checkbox with decimal fields

Let’s artificially modify our application to show how DdsCheckboxField could be used to bind a Decimal field in our Model (which could be extended to a database field defined as decimal too).

To avoid changing too much our Application, let’s add a new field to the Model defined as follows:

\$\SunFarm\CustomerAppSite\Areas\CustomerAppViews\Pages\CUSTDSPF.cshtml.cs:

```
[Dec(1, 0)]
[Values(typeof(decimal), 1, 0)]
public decimal DECSNDCONF { get; set; }
```

The new field reads “Decimal Send Confirmation” and uses 1 digit, where 1 means **Yes** and 0 (zero) means **No**

For convenience we will declare it right after PERCENT_CHANGE_RETURNS (around line 250 - see commit file differences)

¹ Commit “Replaced Y/N field with Checkbox”

In the markup, replace: CUSTREC.SFYN01 with CUSTREC.DECSNDCONF:

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

Before:

```
<DdsCheckboxField Col="47" Text="Send Confirmation" For="CUSTREC.SFYN01" VirtualRowCol="18,27" />
```

After:

```
<DdsCheckboxField Col="47" Text="Send Confirmation" For="CUSTREC.DECSNDCONF"
VirtualRowCol="18,27" />
```

If we had DECSNDCONF in the database declared as decimal (1,0) that would be all we need to do, but we will not go to the trouble of changing the database schema.

Instead, we will trick the Logic code such that Y/N in field SFYN01 will update DECSNDCONF when going out to the screen, and the posted new value for DECSNDCONF with value 1 or 0 will update SFYN01 on the way in. This should be simple to do.

Let's first declare DECSNDCONF in file:

\$\SunFarm\CustomerAppLogic\CUSTINQ.io.cs

```
private decimal DECSNDCONF;
```

(For convenience we will do it after the declaration for field PERCENT_CHANGE_RETURNS, around line 293)

Then we will make sure the database field is bound with the new decimal field, by adding code to the PopulateBuffer and PopulateFields in the IO code for CUSTINQ:

\$\SunFarm\CustomerAppLogic\CUSTINQ.io.cs

```
private void PopulateBufferCUSTDSPFCUSTREC(AdgDataSet _dataSet)
{
    var _table = _dataSet.GetAdgTable("CUSTREC");
    System.Data.DataRow _row = _table.Row;
    .
    .
    .
    _row["DECSNDCONF"] = (SFYN01 == "Y") ? 1 : 0;
}
```

```

private void PopulateFieldsCUSTDSPFCUSTREC(AdgDataSet _dataSet)
{
    var _table = _dataSet.GetAdgTable("CUSTREC");
    System.Data.DataRow _row = _table.Row;

    DECSNDCONF = ((decimal)(_row["DECSNDCONF"]));

    .
    .
    .

    SFYN01 = DECSNDCONF == 1 ? "Y" : "N";
}
2

```

Build CustomerAppLogic project and run the Website.

Updating Customer records by changing the state of the “Send Confirmation” checkbox will work the same way as before, but you can debug to see how the DECSNDCONF decimal field values changes from 0 to 1 values.