Technical Note 102

Implementing an auto-complete choice list

Overview

When dealing with choice lists that contain a large number of choice items, it is important to find a way to manage them efficiently in order to improve the performance, responsiveness and usability of applications. Auto–complete choice lists make their management easier.

An auto–complete choice list is a list that populates itself with choice items that match a substring: the list adapts as characters are typed in an edit box or combo box. The substring may be found at the start, the end or anywhere within the choice item, depending on the SQL command that you provide.

The Rep++ for ASPNET Core framework provides an auto-complete feature that you can use on a choice list built at execution from an SQL command. It is implemented with the Typeahead.js or Select2 libraries, which inject the choice items in the list.

This article describes how to implement the auto-complete feature in your application.[1]

[1] *The products and software cited in this document are registered trademarks, trademarks or trade names of their respective holders.*

Implementing an auto-complete choice list on a field

An auto–complete choice list is a list that populates itself with choice items that match a substring: the list adapts as characters are typed in an edit box or a combo box. The substring may be found at the start, the end or anywhere within the choice item, depending on the SQL command that you provide. The auto-complete functionality may be activated using a Rep++ user-defined attribute in Rep++ Studio.

In this article, you will first implement the auto-complete on a field having a suggested choice list where there is no conversion between internal and displayed values. For these fields, Rep++ provides two types of auto–complete:

- Suggested. In a suggested choice list, the end user can either select an item from the list or type an entry that does not match any existing choice item.
- Validated. In a validated choice list, the end user can only select an item from the list. An entry that does not match any existing item is cleared when the control loses focus.

In the second implementation, the field is of type alphanumeric or numeric choice list, where the displayed values and the saved values are not the same.

Prerequisites for using the auto-complete functionality for a field

The field on which the auto-complete feature is applied must satisfy the following conditions:

- The field must have a choice list (can be suggested).
- The control type associated with the field is an edit box to use the Typeahead.js library, or a combo box to use the Select2 library.
- The field must have an associated SQL command to load the choice list.

Sample system

A sample system designed to demonstrate this capability, Technote102, is included with the Rep++ installation. The information is found in two tables, TN102_CLIENT and TN102_COMPANY. The tables contain the following columns:

Company table

Ciecode	Ciename

Client table

Clientcode	ClientFirstname	Clientlastname	Clienttype	**Ciecode**	Clientsalestodate	...

The auto-complete functionality also requires a set of specially designed attributes provided by Rep++, which you will enable.

The functionality will be tested in a single-page application (SPA) powered by ASP.NET Core and Angular. You will test the auto-complete on two types of fields: a field whose displayed value is the same as the saved value, and a field whose displayed value is different from the saved value.

In summary, you will:

1. Create the Technote102 database tables and import the system in Rep++.
2. Import the user-defined attributes that implement the auto-complete feature.
3. Create a Rep++ ASPNET Core Angular SPA application in Visual Studio® using the Rep++ wizard to test your application.
4. Set up the Clientfirstname field as an auto-complete choice list.
5. Set up the Ciecode field as an auto-complete choice list with a join operation.

Prerequisites

- Rep++ installed (includes Rep++ Studio and the Rep++ connection manager called Tool).
- Working knowledge of Rep++.
- SQL Server as database.

Create the Technote102 database tables and import the system

The tables for this system will be added to the repository for the chosen connection.

To create the Technote102 database tables

1. Open Tool.
2. Double-click the connection where you want to test the auto-complete feature.
3. On the top right, click **Script**.
4. Click **Open** on the script toolbar.
5. Choose the Demo/Technotes/Technote102_Srv.sql file under the Rep++ installation folder.
6. On the toolbar, click **Execute Script**. Your tables have been added.

To import the Technote102 system in your connection

1. In your connection window, on the top right, click **Repository**.
2. In the **Actions** box, click **Import a system**.
3. Choose the Demo/Technotes/Technote102.sys system under the Rep++ installation folder.

The system has been added to your connection. You can close Tool.

Import the user-defined attributes

You may already have imported the FWSPA user-defined attributes in your connection through a previous exercise. If this is the case, skip the import, and include them in the section where you want them, in this case, Technote102.

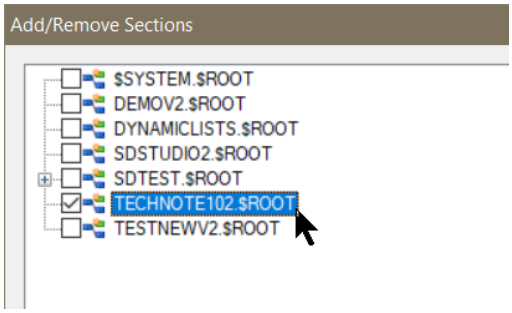
To import the user-defined attributes

1. Open the Rep++ User-Defined Attributes Editor using the connection where you imported the Technote102 system.
2. On the **Tools** menu, click **User-Defined Attributes Import**.
3. In the *User-Defined Attributes Import* dialog box, select the SPAUDADef.uda file located under the Rep++ installation folder, in the Rep/base subfolder. Click **OK**.

The set of user-defined attributes that implement the auto-complete functionality are included in the FWSPA component, which now appears in the components list. To use the auto-complete feature, you must explicitly specify the Rep++ section(s) where the FWSPA user-defined attributes will be available.

To include the user-defined attributes in a section

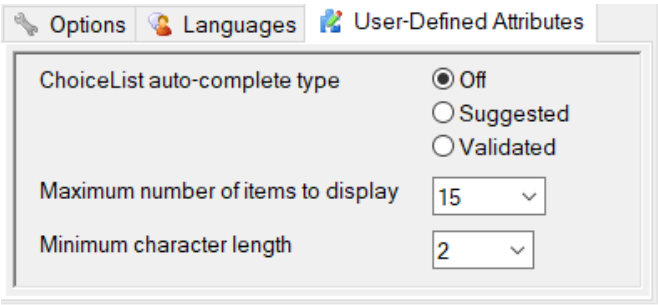
1. In the *User-Defined Attributes* editor, select the FWSPA component and click the **Sections** tab below the definition section.
2. Click **Add/Remove**.
3. Check the section containing the program where you want to use the auto-complete and click **OK**.



4. Save your modifications.
5. Restart Rep++ studio for the changes to take effect.

The auto-complete attributes

The auto-complete attributes are added in the Rowset editor at the field level. To view the attributes, open the Rowset editor and click the **User-Defined Attributes** tab below the fields list. The attributes values are originally set to null.



The attributes are described below.

Choice list auto-complete type	Sets the behaviour of the auto-complete feature. **Off**. Disables the auto-complete feature. **Suggested**. Enables the auto-complete feature. The end user is allowed to choose a from the list or type text that does not match any existing choice item. **Validated**. Enables the auto-complete feature. The end user can only save a choice item from the list: Text that does not match any existing item is cleared when the control loses focus.
Maximum number of items to display	Maximum number of items that is proposed to the end user. This limits the number of matching items returned. A value of 0 indicates that all values will be displayed.
Minimum character length	Number of characters that triggers a call to the auto-complete function. Must be 1 or greater: a value of 3 is reasonable with large lists.

Create a Rep++ ASP.NET Core application to test your functionality

To test your new functionality, create a Rep++ ASP.NET Core application using the Rep++ wizard.

To create a Rep++ ASP.NET Core application using the Rep++ wizard

1. In Visual Studio®, create a Rep++ V9 ASP.NET Core Application.
2. In the Rep++ wizard's *Connect to the Rep++ repository* page, select the connection, system and program where Technote102 is located.
3. In the Select *Rep++ repository components* page, select the *ClientTransaction* and *ClientSelectTransaction* RowsetTrees for the transaction and selection buffer, respectively.
4. In the *Select n-tier options* page, leave the default values.
5. In the *Specify generation information for typed instances* page, leave the default values.
6. In the *Specify generation information for POCO entities* page, clear **Generate POCO entities**.
7. In the *Select template* page, select *ASPNET Core Single Page Application Angular Template* in the **Available Templates** list.
8. In all subsequent steps, leave the default values then click **Finish** to create the application.

Set up the Clientfirstname field as an auto-complete choice list

In this first example, you will implement the auto-complete function on the CLIENTFIRSTNAME field: you will display a list of the existing first names.

You first need to activate the auto-complete function for the CLIENTFIRSTNAME list, define the SQL command to build the list, then modify the settings of the field within the Rowset, and finally test your program.

Figure 1. Auto complete using typeahead.js, for edit box controls.

Figure 2. Auto complete using Select2, for combo box control.

To set up the Clientfirstname choice list

1. In Rep++ studio, open the Technote102 system, then the Technote102 program.
2. Open the CLIENT Rowset.
3. Select the CLIENTFIRSTNAME field from the list.
4. Click the **User-Defined Attributes** tab below the **Fields** list.
5. Set the attributes as follows:
   - Choice list auto-complete type: **Suggested** or **Validated**
   - Maximum number of items to display: **15**
   - Minimum character length: **1**
6. Click the **Options** tab below the **Fields** list.
7. Click the label **SQL Command for List**.

8. In the *SQL command editor* window, create a SQL command named *List_Clientfirstname* with the following instructions:

```
SELECT DISTINCT CLIENTFIRSTNAME
FROM TN102_CLIENT
WHERE
CLIENTFIRSTNAME LIKE SD_CONCAT(SD_CONCAT('%',:K_OptStrParamAutoComplete),'%')
ORDER BY CLIENTFIRSTNAME
```

   Note: *K_OptStrParamAutoComplete* is a variable that will contain the auto complete search term as the end user types it in the field.

9. Save the SQL command and close the editor.
10. In the **Options** page, next to **SQL Command for list**, select *List_Clientfirstname*.
11. Save your modifications, and launch the Rep++ SPA application to test your changes.

Notice the difference when you change the type of the auto-complete from **Suggested** to **Validated** and the control type from **edit box** to **combo box**.

Set up the Ciecode field as an auto-complete choice list

There are situations when the auto-complete field displays choice items having an internal value, typically not user-friendly, to which is associated a more readable display value (external value).

In the excerpt below, the TN102_COMPANY table contains the CIECODE and CIENAME columns:

	CIECODE	CIENAME
1	ALP	Alpha Inc.
2	BETA	Beta corp.
3	CBE	Quebec Blue Eyes Community
4	CONSYST	Consyst SQL Inc.
5	DEL	Delta Delivery Inc.
6	ENE	Energyol Solutions
7	FIM	ForInstance Marketing
8	GAM	Gamma Soup
9	GES	Gestalt HR
10	HAB	Habit 4U

The company information is included through the CIECODE field in the Rowset. You want, however, your end users to choose from CIENAME, as in:

When the internal and external values are different, the auto-complete feature is set up differently:

1. The auto-complete field must be of type **Alpha simple choice** or **Numeric simple choice**.
2. Its control type must be set to **Combo box**.

3. The auto-complete type of the field must be set to **Validated**.
4. The automatic refresh option of the field must be enabled.
5. Its associated SQL Command must do two things:
   a. Load the list of choices as the user types.
   b. Load a single choice corresponding to the internal value when data is read from the database

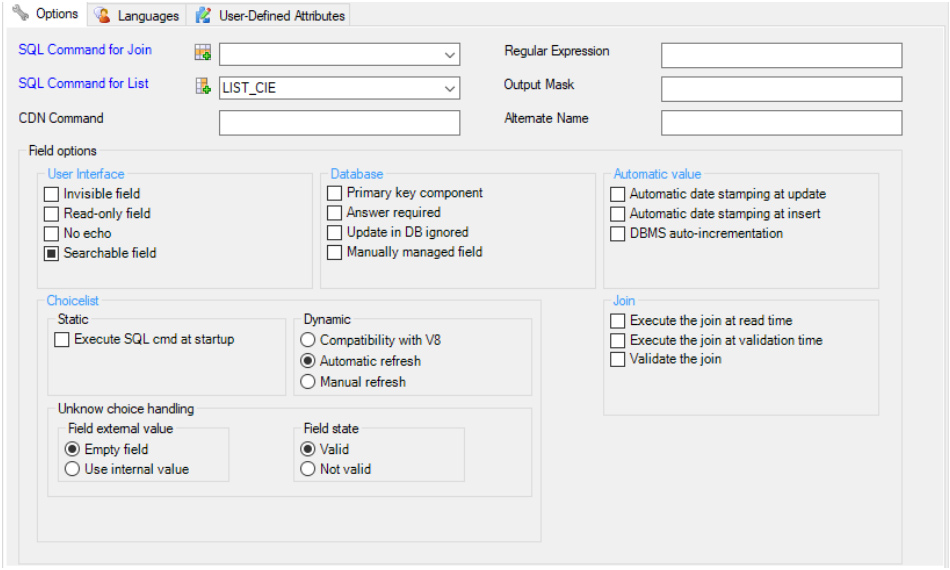
The following example demonstrates how to set up the auto-complete for the CIECODE field in the Technote102 program: the end user will be presented with the company name, but the saved field value is the company code.

To set up the Ciecode field as an auto-complete choice list with a join operation

1. In Rep++ studio, open the Technote102 hierarchy.
2. Click the **Fields** node.
3. Select the CIECODE field from the list. Make sure the following attributes are set:
   - Column: **CIECODE**
   - Type: **Alpha simple choice**
   - Control Type: **Combo box**
   - Max size: **16**
4. Under the TECHNODE102 program node, click the **Rowsets** node and select the CLIENT Rowset.
5. Select the CIECODE field from the list.
6. In the **Options** page below the list, check the **Automatic refresh** option in the *Dynamic* choice list group.
7. Click the **User-Defined Attributes** tab below the fields list and set the attributes as follows:
   - ChoiceList auto-complete type: **Validated**
   - Maximum number of items to display: **10**
   - Minimum character length: **1**
8. Click the **Options** tab.
9. Click the label **SQL Command for List** to open the SQL command editor.
10. Create the LIST_CIE command. The SQL instructions appear below.

```
SELECT CIECODE, CIECODE, CIENAME
FROM TN102_COMPANY
WHERE
(:K_OptStrParamAutoComplete IS NULL AND
 CIECODE = :CLIENT.CIECODE)
OR
(:K_OptStrParamAutoComplete IS NOT NULL AND
 CIENAME LIKE SD_CONCAT(SD_CONCAT('%',:K_OptStrParamAutoComplete),'%'))
ORDER BY CIENAME
```

11. Back in the Rowset editor, select LIST_CIE from the list **SQL Command for List**. The **Options** page for the CIECODE field should look like this.



12. **Optional.** Click the **Languages** tab and type ***Company*** as the prompt in the English language.
13. Save your Rowset.
14. Launch the Rep++ SPA application to test your changes.