Technical Note 103

Triggering an automatic choice list rebuild in Rep++ single-page applications

Overview

In a Web context, it is necessary to maintain a pool of already initialized Rep++ Application objects in order to minimize delays. Dynamic choice lists are however built when an Application object is initialized. This means choice lists for existing objects become outdated when changes are made to the tables from which the choice lists are built.

The Application objects must therefore be made aware that a change occurred and that their choice lists must be rebuilt when they are retrieved from the pool.

This article will demonstrate how to refresh dynamically built choice lists in a Rep++ single-page application (SPA).

Triggering an automatic choice list rebuild in Rep++ single-page applications

In a Web context, it is necessary to maintain a pool of already initialized Rep++ Application objects in order to minimize delays. Dynamic choice lists are however built when an Application object is originally initialized. This means choice lists for existing objects become outdated when changes are made to the tables from which the choice lists are built.

The Application objects that live in the pool must therefore be made aware that a change occurred in order to rebuild their choice lists when necessary.

Therefore, to notify existing Application objects, two things need to happen:

- A program that modifies tables from which choice lists are built must be able to indicate that changes occurred.
- An Application object retrieved from the pool must be able to check if its choice lists are outdated and perform a refresh accordingly.

The Rep++ framework for SPA transparently associates a ChoiceListUpdater object with each application it creates. The ChoiceListUpdater class (in the RepPP.Toolkit.SPA namespace) provides the methods and properties to trigger a refresh of dynamic choice lists: an application that modifies the content of tables can call a method that outdates choice lists, while an Application object retrieved from the pool can query whether its choice lists need to be refreshed.

In this article, you will demonstrate that a dynamic choice list is properly refreshed. To that end, you will create two applications from the same system, one to manage companies, the other to manage clients. The company management application will call a method that outdates choice lists every time modifications to the companies are saved. The client management application contains a dynamically built list of companies; it automatically checks if a rebuild is necessary. You will start both applications, modify the companies list in the company application, then see how the list of the client application is refreshed.

Sample system

A sample system designed to demonstrate this capability, Technote103, is included with the Rep++ installation. The information is found in 2 tables, TN103_CLIENT and TN103_COMPANY. The tables contain the following columns:

Company table

Ciecode	Ciename

Client table

Clientcode	ClientFirstname	Clientlastname	Clienttype	Ciecode	Clientsalestodate	...

The system also defines 4 RowsetTrees:

- ClientTransaction and ClientSelectTransaction, used in the client management application.
- CompanyTransaction and CompanySelectTransaction, used in the company management application.

You do not need make any modifications to the system: the Rowsets and SQL command have already been included. The refreshing functionality is done programmatically. It will be tested in two single-page applications (SPAs).

In summary, you will:

1. Create the technote's database tables and import the system.
2. Create a Rep++ SPA application for the company management application.
3. Integrate the code for refreshing the company list.
4. Create a Rep++ SPA application for the client management application.
5. Test your application.

Prerequisites

Rep++ installed (includes Rep++ studio and SD Tools). Working knowledge of Rep++. SQL Server as database.

Create the Technote103 database tables and import the system

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

To create the Technote103 database tables

1. Open SD Tools.
2. Double-click the connection where you want to test the feature. A window with your connection contents opens.
3. On the **File** menu, click **Open**.
4. Choose the Demo/Technotes/Technote103_Srv.sql file under the Rep++ installation folder.
5. On the toolbar, click **Execute**.

To import the Technote103 system in your connection

1. In your open connection window, on the right pane, double-click **Import a system**.
2. Open the Demo/Technotes/Technote103.sys system under the Rep++ installation folder.

The system has been added to your connection. You can close SD Tools. You may open Rep++ studio to take a look at the components, but no modifications will be made to the system.

Create the Rep++ SPA company management application

To create a Rep++ SPA company management application using the Rep++ wizard

1. In Visual Studio®, create a standard ASP.NET Web Application (not Rep++) named TN103_Company.
2. In the *New ASP.NET Project* window, select the **Rep++ MVC V8** template.
3. In the Rep++ wizard's *Connect to the Rep++ repository* page, select the connection, system and program for technote TN103.
4. In the Select *Rep++ repository components* page, select the *CompanyTransaction* and *CompanySelectTransaction* RowsetTrees for the transaction and selection buffer, respectively.
5. In the *Select n-tier options* page, leave the default values.
6. In the *Specify generation information for typed instances* page, leave the default values.
7. In the *Specify generation information for POCO entities* page, clear **Generate POCO entities**.
8. In the *Select template* page, select *MVC SPA using Angle Template*.
9. In the *Select layout options* leave the default values.
10. In the *Select replacement mode* page, click **Finish** to create the application.
11. In the *Search for TypeScript Typings* message box, click **No**.

Integrate the code for refreshing the company list

It is the programmer's responsibility to indicate that whenever the COMPANY table is modified, all choice lists that refer it should be rebuilt. To that end, the company management application will call the SPAController.TouchChoiceList method (in the RepPP.Framework.SPA namespace) every time modifications to the companies are saved. This method helps other applications determine if their dynamic choice lists are still valid or outdated.

You'll override the SaveTransactionToDataAccess method of the SPA controller so that when modifications to the company list are saved, the TouchChoiceList method is called.

To integrate the code for refreshing the company list

1. Open the CompanyTransactionsController.cs file.
2. Add the following directive:

```
using System.Collections.Generic;
```

3. Override the SaveTransactionToDataAccess method by adding the following code:

```
protected override int SaveTransactionToDataAccess(RowsetTree rstTrans,
    bool bAllLines, Dictionary<string, string> userParameters) {

    int retVal;

    retVal = base.SaveTransactionToDataAccess(rstTrans,
                                              bAllLines,
                                              userParameters);
    TouchChoiceList();
    return retVal;
}
```

4. Build and run your application.

Create the Rep++ SPA client management application

To create a Rep++ SPA client management application using the Rep++ wizard

1. In Visual Studio®, create a standard ASP.NET Web Application (not Rep++) named TN103_Client.
2. Follow the steps above, but in the Rep++ wizard's *Connect to the Rep++ repository* page, use the *ClientTransaction* and *ClientSelectTransaction* RowsetTrees.
3. Build and run your application.

The ChoiceListUpdater object in the client management application seamlessly determines if a refresh is required, and if so, rebuilds the choice lists. No particular code is required here, it is inherently implemented by the framework.

Test the feature

To test the feature

1. In the TN103_Client application, open the company list and look at its content.
2. In the TN103_Company application, modify the company list by adding, deleting or changing a company.
3. In the TN103_Client application, click the refresh button of the browser, open the company list on any client, and see that the content has changed accordingly.

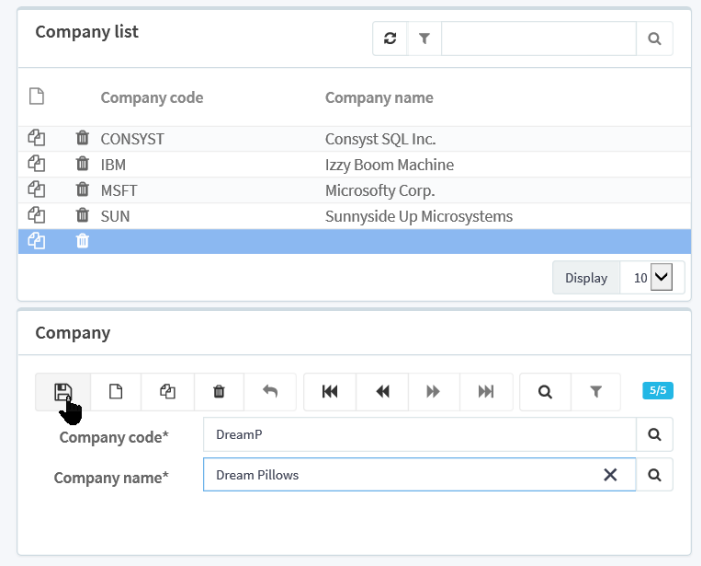

Figure 1. Adding a new company to the TN103_Company application.

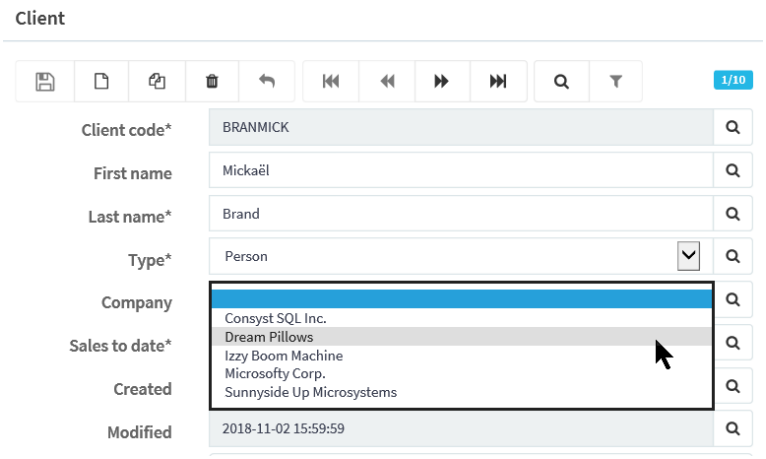

Figure 2. After a refresh of the TN103_Client application.