单元格值改变时更新单元格类型插件

1. 描述

在一些情况下，单元格值变化后，单元格类型需要变更。比如在树型图插件中，如果查询条件变更，则树型图的节点也需要更新。

需要使用IDependenceCells接口来实现插件动态数据更新。

2. 单元格值改变时更新单元格类型插件

操作步骤

1. 实现接口IDependenceCells ，并列举所有有依赖的单元格。

```
namespace MyListCellType
{
    [Designer("MyListCellType.MyListCellTypeDesigner,MyListCellType")]
    public class MyListCellType : CellType
    {
        public string TableName
        {
            get; set;
        }
        public string TextColumn
        {
            get; set;
        }
        [FormulaProperty]
        public object SimpleQueryCell
        {
            get; set;
        }
    }

    public class MyListCellTypeDesigner :
CellTypeDesigner<MyListCellType>, IDependenceCells
    {
        public override EditorSetting
GetEditorSetting(PropertyDescriptor property, IBuilderContext
builderContext)
        {
            if (property.Name == "TableName")
            {
                return new TableComboTreeSelectorEditorSetting();
            }
            if (property.Name == "TextColumn")
```

```
            {
                var columns =
builderContext.EnumAllTableInfos().FirstOrDefault(t => t.TableName ==
this.CellType.TableName)?.Columns?.Select(c => c.ColumnName);
                return new ComboEditorSetting(columns);
            }
            if (property.Name == "ValueColumn")
            {
                var columns =
builderContext.EnumAllTableInfos().FirstOrDefault(t => t.TableName ==
this.CellType.TableName)?.Columns?.Select(c => c.ColumnName);
                return new ComboEditorSetting(columns);
            }
            return base.GetEditorSetting(property, builderContext);
        }

        public override FrameworkElement GetDrawingControl(ICellInfo
cellInfo, IDrawingHelper drawingHelper)
        {
            ListBox listBox = new ListBox();
            //get table data for preview.
            var tableData =
drawingHelper.GetTableDataForPreview(this.CellType.TableName, new
List<string>() { this.CellType.TextColumn }, null, true);
            if (tableData != null)
            {
                foreach (var row in tableData)
                {
                    var value = row[this.CellType.TextColumn];
                    if (value != null)
                    {
                        ListBoxItem item = new ListBoxItem() { Content =
value };
                        listBox.Items.Add(item);
                    }
                }
            }
            Grid container = new Grid();
            container.Children.Add(listBox);
            return container;
        }

        public IEnumerable<object> EnumDependenceCells(IBuilderContext
context)
        {
            return
context.EnumDependenceCellsFromFormula(this.CellType.SimpleQueryCell);
```

```
        }
    }
}
```

在这个示例中，您可以使用IBuilderContext.EnumDependenceCellsFromFormula(object formula)来列举所有有依赖的单元格，您还可以使用IBuilderContext.EnumDependenceCells(object queryCondition)来列举所有有依赖的单元格。

2

添加onDependenceCellValueChanged回调函数如下，并重新加载列表数据。

```
var MyListCellType = (function (_super) {
    __extends(MyListCellType, _super);
    function MyListCellType() {
        return _super !== null && _super.apply(this, arguments) || this;
    }

    MyListCellType.prototype.createContent = function () {
        var self = this;
        var element = this.CellElement;
        var container = $("<div id='" + this.ID + "'></div>");
        var listCell = $("<ul id='" + this.ID + "_list' ></ul>");
        listCell.css("width", element.Width + "px");
        listCell.css("height", element.Height + "px");
        //
        this.loadItems();
        container.append(listCell);

        this.onDependenceCellValueChanged(function () {    //add
callback event and reload the list items
            //clear children
            $("#" + self.ID + "_list").empty();
            //
            self.loadItems();
        });
        return container;
    }

    MyListCellType.prototype.loadItems = function () {
        var self = this;
        var element = this.CellElement;
        var cellTypeMetaData = element.CellType;
        var dependenceCell = cellTypeMetaData.SimpleQueryCell;
        var dependenceValue = self.evaluateFormula(dependenceCell);
        var odataParam = cellTypeMetaData.TableName + "?$select=" +
cellTypeMetaData.TextColumn;
        if (dependenceValue != null && dependenceValue !== "") {
            odataParam += "&$filter=contains(" +
cellTypeMetaData.TextColumn + ",'" + dependenceValue + "')";
        }
        //
        Forguncy.getTableDataByOData(odataParam, function (tableData) {
            if (tableData) {
                for (var row = 0; row < tableData.length; row++) {
                    var rowData = tableData[row];
                    var text = rowData[cellTypeMetaData.TextColumn];
                    var li = $("<li>" + text + "</li>");
```

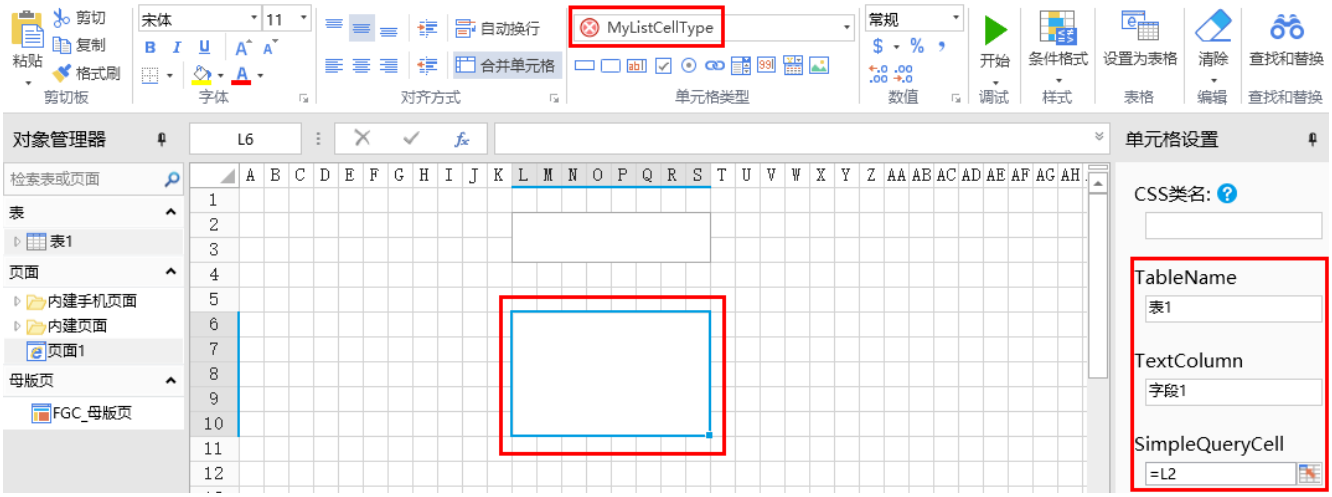
```
                    $("#" + self.ID + "_list").append(li);
                }
            }
        });
    }

    return MyListCellType;
}(Forguncy.CellTypeBase));
```

```
// Key format is "Namespace.ClassName, AssemblyName"
Forguncy.Plugin.CellTypeHelper.registerCellType("MyListCellType.MyListCellType, MyListCellType", MyListCellType);
```

结束

重新构建工程并重启设计器，设置依赖单元格如下：



运行页面，当L2单元格值变化时，列表也会更新。

- 当L2没有输入值时，显示如下：

  - a1
  - a2
  - b1
  - b2
  - b3

- 当L2输入“a”时，显示如下：

  - a1
  - a2

- 当L2输入“b”时，显示如下：

b

- b1
- b2
- b3

回到顶部