导出单元格值到Excel

1.描述

如果单元格类型插件需要导出到Excel，单元格类型需要实现IExportCellType接口。

查看完整代码请参见：https://gitee.com/huozige-china/list-cell-type。

2.导出单元格值到Excel

下面的示例中，当导出页面到Excel时只导出单元格的值，并只导出显示的文本。

操作步骤

1. 在.cs文件中，添加如下代码：

```
namespace MyListCellType
{
    [Designer("MyListCellType.MyListCellTypeDesigner,MyListCellType")]
    public class MyListCellType : CellType, IExportCellType
    {
        public string TableName
        {
            get; set;
        }
        public string TextColumn
        {
            get; set;
        }
        public string ValueColumn
        {
            get; set;
        }
        public bool ExportPicture
        {
            get
            {
                return false;
            }
        }

        public ExportResultInfo ExportToExcel(ICellInfo targetCell,
IExportContext context)
        {

            var result = new ExportResultInfo();
            var cellValue = targetCell.Value;    //get cell value.
```

```
            if (cellValue != null)
            {
                var queryInfo = new Dictionary<string, object>();
                queryInfo.Add(ValueColumn, cellValue);
                //find the display text using cellValue from table data
                var tableData =
context.ValueProvider.GetTableData(TableName, new List<string>() {
TextColumn }, new List<string>() { "ID" }, queryInfo, true);    //get
the table row data when the ValueColumn equals to the cell value.
                if (tableData != null && tableData.Count > 0 &&
tableData[0].ContainsKey(TextColumn))
                {
                    var displayText = tableData[0][TextColumn];
                    result.ExportValue = displayText;    //export the
display text only, and you also can export value if you change these
code.
                    return result;
                }
            }

            result.ExportValue = null;
            return result;
        }
    }


    public class MyListCellTypeDesigner :
CellTypeDesigner<MyListCellType>
    {
        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;
        }
    }
}
```

在这个示例中，您可以从 ICellInfo获取单元格值，也可从IExportContext.ValueProvider获得数据表的数据。

2

在.js文件中，添加如下代码：

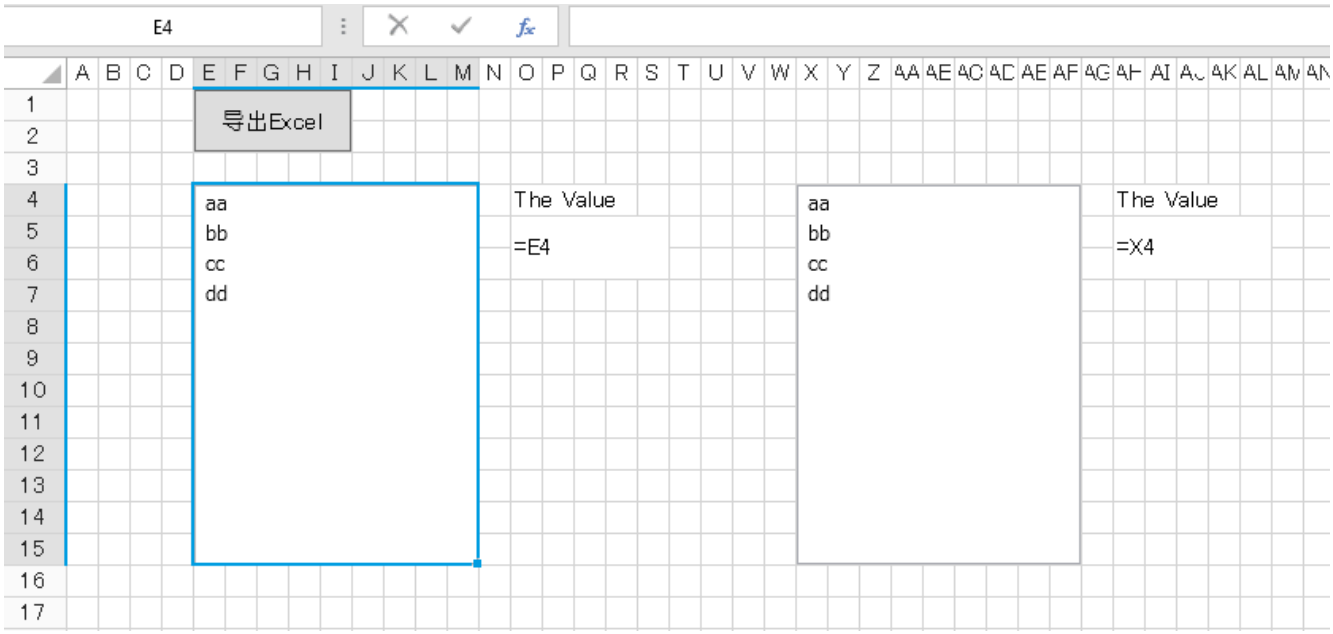
```
var MyListCellType = (function (_super) {
    __extends(MyListCellType, _super);
    function MyListCellType() {
        return _super !== null && _super.apply(this, arguments) || this;
    }
    MyListCellType.prototype.myValue = null;

    MyListCellType.prototype.createContent = function () {
        var self = this;

        var element = this.CellElement;
        var cellTypeMetaData = element.CellType;
        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 tableName = cellTypeMetaData.TableName;
        var columns = [cellTypeMetaData.TextColumn];
        if (columns.indexOf(cellTypeMetaData.ValueColumn) === -1) {
            columns.push(cellTypeMetaData.ValueColumn);
        }
        var queryCondition = cellTypeMetaData.QueryCondition;

        var param = {
            TableName: tableName,   //
```

```
            Columns: columns,    //
            QueryCondition: queryCondition,    //
            QueryPolicy: {
                Distinct: true,
                QueryNullPolicy:
Forguncy.QueryNullPolicy.QueryAllItemsWhenValueIsNull,
                IgnoreCache: false
            },
            SortCondition: null    //
        };

        var formulaCalcContext = {
            IsInMasterPage: false   //
        };

        Forguncy.getTableDataByCondition(param, formulaCalcContext,
function (data) {
            var tableData = data;
            for (var row = 0; row < tableData.length; row++) {
                var text = tableData[row][cellTypeMetaData.TextColumn];
                var value =
tableData[row][cellTypeMetaData.ValueColumn];
                var li = $("<li value = " + value + ">" + text +
"</li>");

                li.on("click", function () {
                    self.myValue = $(this).attr("value");
                    $(this).parent().find("li").css("color", "black");
                    $(this).css("color", "red");
                    self.commitValue();
                });
                $("#" + self.ID + "_list").append(li);
            }
        }, true);
    };

    MyListCellType.prototype.getValueFromElement = function () {
        return this.myValue;
    };

    MyListCellType.prototype.setValueToElement = function (element,
value) {

    };

    MyListCellType.prototype.disable = function () {
        _super.prototype.disable.call(this);
    };

    MyListCellType.prototype.enable = function () {
        _super.prototype.enable.call(this);
    };

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

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

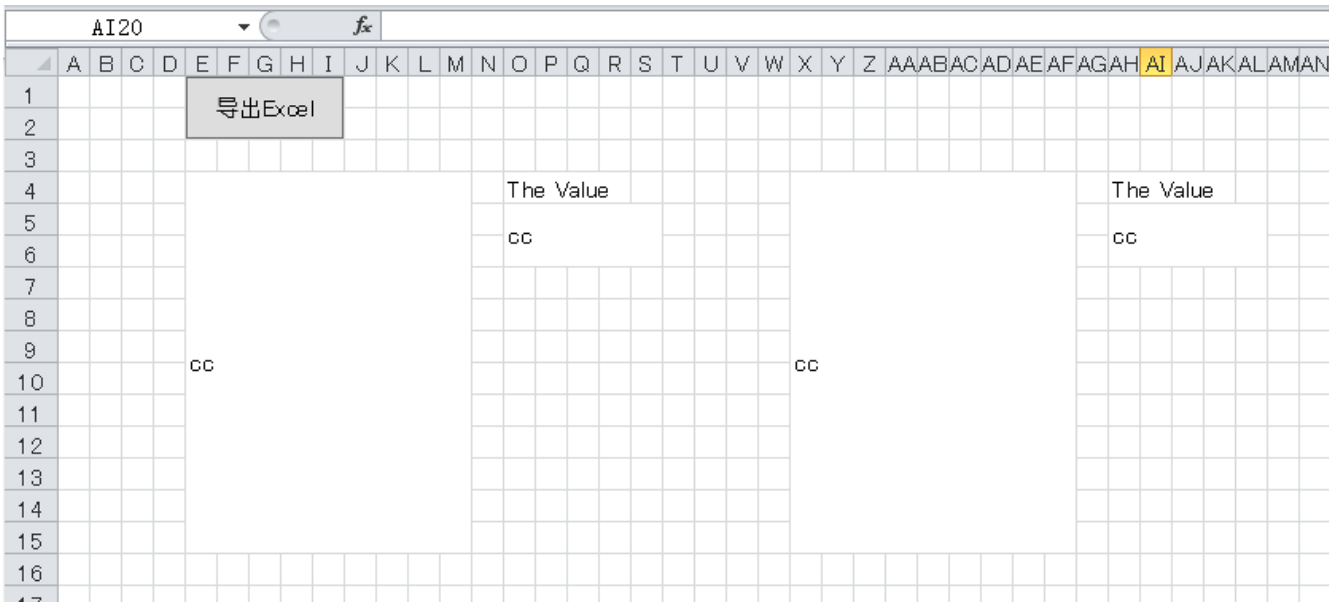
结束

重新构建工程并重启设计器，在设计器页面中设置单元格类型为MyListCellType ，第一个单元格中文本和值相同，第二个单元格中文本和值不同。



运行页面，在这两个单元格中都单击“cc”，您会看到值是不同的，如下所示：

单击“导出Excel”按钮，页面会导出到Excel，如下所示：



回到顶部