# Evaluator的说明

Evaluator是借鉴Eval Expression.NET早起版本的一个轻量级表达式计算。

它支持大多数的常规C#语法，需要注意以下几个问题：

* 字符串用单引号'，不要使用双引号"
* 不支持类的索引器，不如row["field"]这种语法方式，因为它认为[]都是数组，所以碰到上述情况，建议包装方法进行访问。
* 不要尝试在表达式中使用扩展方法，Linq等高深的语法

## Evaluator原生支持类型

Evaluator原生支持如下类型：

* bool
* byte
* char
* int
* decimal
* double
* float
* object
* string
* DateTime
* Convert
* Math

以上均对应System中对应的类型。请注意，在表达式中必须要如此写，并注意大小写敏感的问题。

除上述类型以及它们的成员函数外，如果需要调用，均需要外部传入，只有外部传入后方能调用。

比如，下列方法就传入value和row，可以在表达式中调用

return EvaluatorUtil.Execute<bool>(expression,

Tuple.Create("value", value), Tuple.Create("row", rowValue));

## Evaluator扩展支持类型

上面表达式中，由于value，row都有上下文必须根据实际需要传入外。还有一些类型，比如DataSetExtension静态类，对DataSet，DataTable和DataRow的方法都有很好的封装。此外WebGlobalVariable可以访问Request.QueryString，Session等实际需求比较大的变量。这些都是静态类型，与上下文无关。

按照传统的方式，必须在调用时，显示添加上述类型，才能在表达式中调用，为了减少不必要的代码，同时提高扩展性，系统提供了EvaluateAdditionAttribute，只需要在相关的类型上声明，该类型就会注入表达式。

例如：

[EvaluateAddition(Author = "YJC", CreateDate = "2018-04-16",

Description = "添加DataSetExtension类型扩展")]

public static class DataSetExtension

如此标注后，表达式中就可以使用诸如：

DataSetExtension.GetString(workflowRow, 'CustomData') == '1'

当前系统加注的类型如下：

* DataSetExtension
* FieldValueProviderHelper
* WebGlobalVariable

如有需要，请及时提出，可以再相关类型上加注。

## EvaluatorDisplay和ConditionDisplay

EvaluatorDisplay和ConditionDisplay都使用Evaluator进行表达式计算。除了原有的类型外，还有value（当前值）和row（当前行）两个上下文变量传入。

需要注意的是，为了克服DataRow和Object两种类型的差异，Toolkit引入的IFieldValueProvider来解决上述问题，因此row已经不是原先的DataRow类型了，而是IFieldValueProvider接口（包装了DataRow），所以在Display中使用表达式需要格外注意。

此外，根据验证，Evaluator对DBNull类型支持不太好，因此，建议不要再Evaluator中使用DBNull，如果要检测空，可以考虑ToString()后和''进行比较，相等可以认为是DBNull。

附：

由于Evaluator的功能比较强大，克服了原有宏的缺陷，因此，新作的功能将优先考虑Evaluator，而不是宏。此外，未必Toolkit的版本中，将考虑使用Evaluator来完全替代宏