支持结构类型的 返回结果

从上一节支持返回结果中，我们已经了解到，通过给属性标注ResultToPropertyAttribute，可以在命令执行后生成一个或多个返回结果，以便后续命令使用。

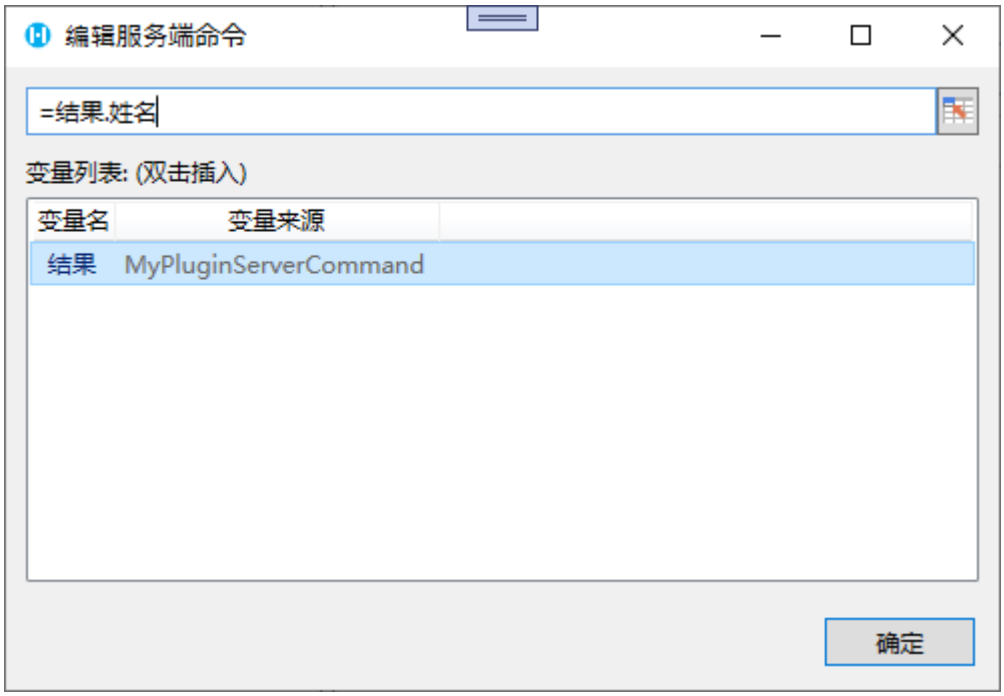
如果希望生成复杂的对象类型返回结果，或者生成数组类型的返回结果，通过标注ResultToPropertyAttribute，也是可以实现的。但是再后续的属性提示中，用户无法快捷的了解返回的对象有哪些子属性。只能通过点操作符硬取值。

例如以下代码：

```
using GrapeCity.Forguncy.Commands;
using GrapeCity.Forguncy.Plugin;
using GrapeCity.Forguncy.ServerApi;
using System.ComponentModel;
using System.Threading.Tasks;

namespace MyPlugin
{
    public class MyPluginServerCommand : Command,
ICommandExecutableInServerSideAsync
    {
        [FormulaProperty]
        [DisplayName("Id")]
        public object StudentId { get; set; }

        [ResultToProperty]
        [DisplayName("")]
        public string ResultTo { get; set; } = "";

        public async Task<ExecuteResult>
ExecuteAsync(IServerCommandExecuteContext dataContext)
        {
            var id = await
dataContext.EvaluateFormulaAsync(this.StudentId);
            var studentInfo = dataContext.DataAccess.GetTableData("",
new ColumnValuePair()
            {
                ColumnName = "ID",
                Value = id
            });

            var result = new ExecuteResult();
            result.ReturnValues.Add(this.ResultTo, studentInfo);
            return result;
        }

        public override CommandScope GetCommandScope()
        {
            return CommandScope.ExecutableInServer;
        }
    }
}
```

对象类型返回值

假设命令执行后，返回学生对象，包含姓名和年龄属性。但是再后续命令使用结果时只会提示结果变量，而如果需要获取子属性值，必须用户手动准确输入，用户体验较差。



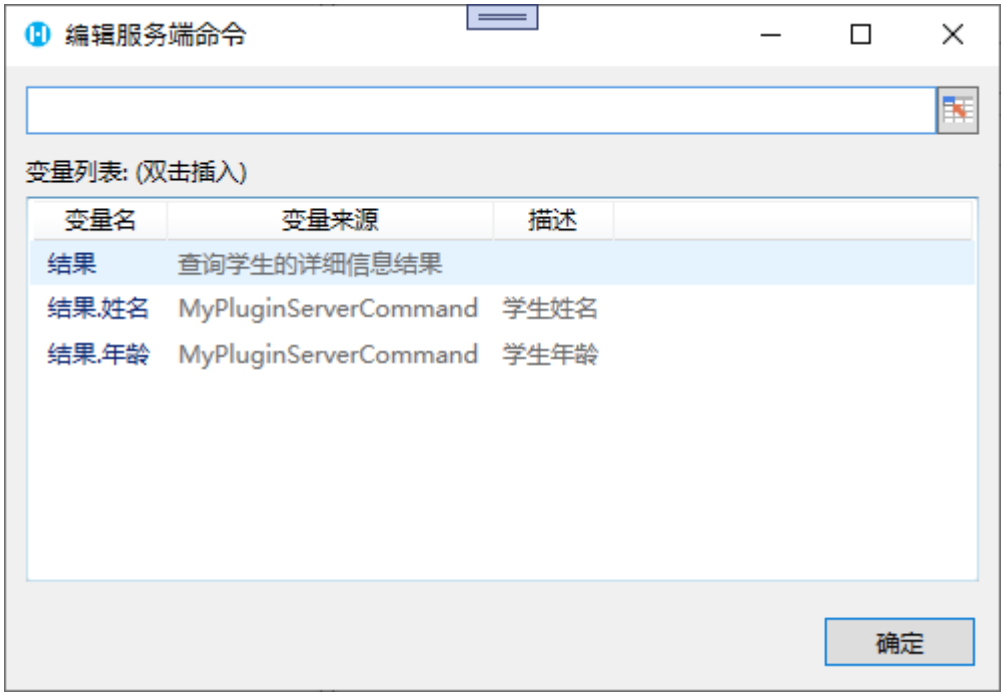
如果希望同时提示子属性，可以通过实现IServerCommandParamGenerator接口实现。

```
using GrapeCity.Forguncy.Commands;
using GrapeCity.Forguncy.Plugin;
using GrapeCity.Forguncy.ServerApi;
using System.Collections.Generic;
using System.ComponentModel;
using System.Threading.Tasks;

namespace MyPlugin
{
    public class MyPluginServerCommand : Command,
ICommandExecutableInServerSideAsync, IServerCommandParamGenerator
    {
        [FormulaProperty]
        [DisplayName("Id")]
        public object StudentId { get; set; }

        [ResultToProperty]
        [DisplayName("")]
        public string ResultTo { get; set; } = "";

        public IEnumerable<GenerateParam> GetGenerateParams()
        {
            yield return new GenerateObjectParam()
            {
                ParamName = this.ResultTo,
                Description = "",
                ParamScope = CommandScope.All,
                SubPropertiesDescription = new Dictionary<string,
string>() {
                    { "",""},
                    { "",""}
                }
            };
        }
```

```
        public async Task<ExecuteResult>
ExecuteAsync(IServerCommandExecuteContext dataContext)
        {
            var id = await
dataContext.EvaluateFormulaAsync(this.StudentId);
            var studentInfo = dataContext.DataAccess.GetTableData("",
new ColumnValuePair()
            {
                ColumnName = "ID",
                Value = id
            });

            var result = new ExecuteResult();
            result.ReturnValues.Add(this.ResultTo, studentInfo);
            return result;
        }

        public override CommandScope GetCommandScope()
        {
            return CommandScope.ExecutableInServer;
        }
```

```
        }
    }
    }
```

效果如下：



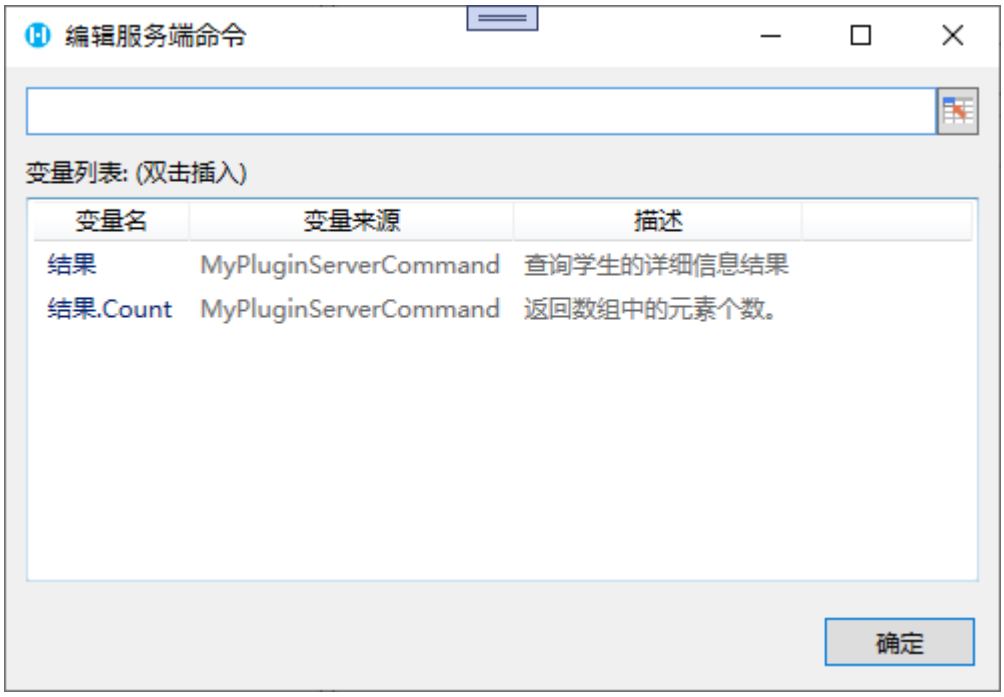
数组类型返回值

如果返回值是列表类型，同样可以通过实现IServerCommandParamGenerator解决。

```
using GrapeCity.Forguncy.Commands;
using System.Collections.Generic;
using System.ComponentModel;
using System.Threading.Tasks;

namespace MyPlugin
{
    public class MyPluginServerCommand : Command,
ICommandExecutableInServerSideAsync, IServerCommandParamGenerator
    {
        [ResultToProperty]
        [DisplayName("")]
        public string ResultTo { get; set; } = "";

        public IEnumerable<GenerateParam> GetGenerateParams()
        {
            yield return new GenerateListParam()
            {
                ParamName = this.ResultTo,
                Description = "",
                ParamScope = CommandScope.All,
                ItemProperties = new List<string>() {
                    "",
                    ""
                }
            };
        }
        public async Task<ExecuteResult>
ExecuteAsync(IServerCommandExecuteContext dataContext)
        {
            var studentInfos = dataContext.DataAccess.GetTableData("");
            var result = new ExecuteResult();
            result.ReturnValues.Add(this.ResultTo, studentInfos);
            return result;
        }

        public override CommandScope GetCommandScope()
        {
            return CommandScope.ExecutableInServer;
        }
    }
}
```

效果如下。

在普通命令中使用时：



在循环命令的子命令中使用时：