插件开发流程

1.描述

活字格提供一个工具，帮助您生成一个模板插件工程，添加您的C#或者JavaScript代码后就可以完成您自定义的单元格类型或者命令插件。

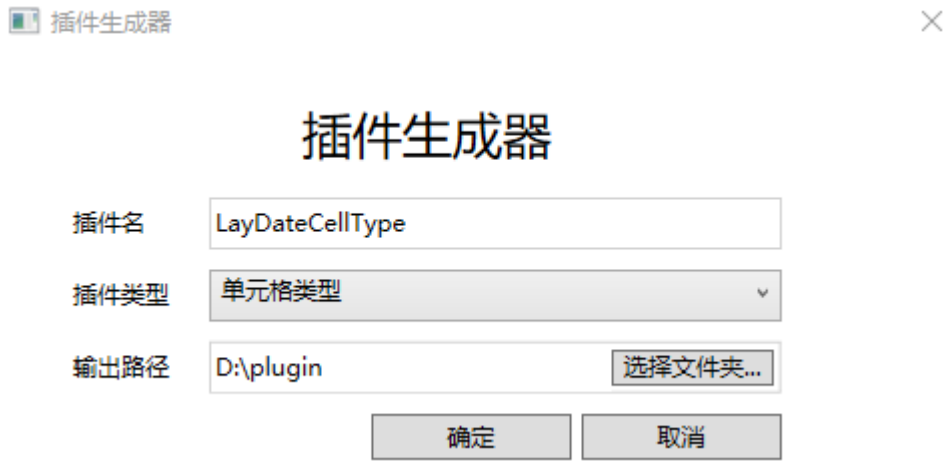
以创建一个日期与时间的单元格类型的插件为例，来讲解开发插件的流程。

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

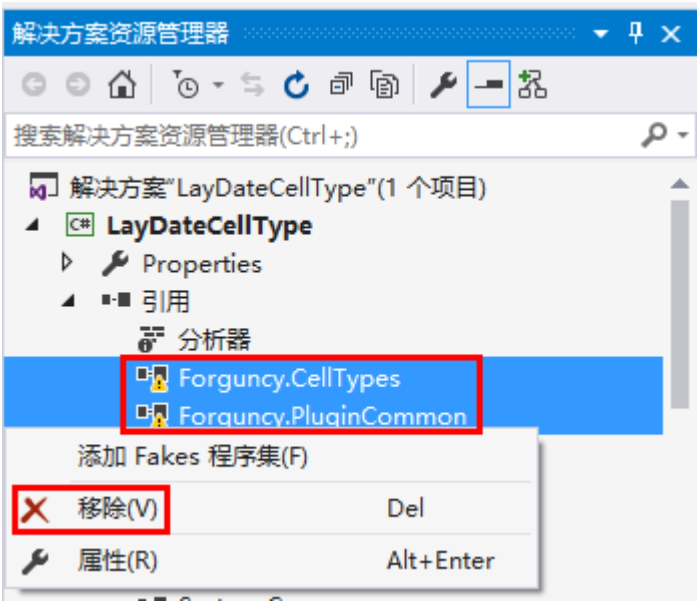
2.插件开发

操作步骤

1. 运行ForguncyPluginCreator.exe，在弹出的对话框中，输入您的插件名称，选择插件类型为单元格类型或命令，并设置插件输出的路径。设置完成后，单击“OK”。



2. 使用Visual Studio打开.csproj文件。
3. 在解决方案资源管理器中将Forguncy.CellTypes和Forguncy.PlugunCommon移除掉。



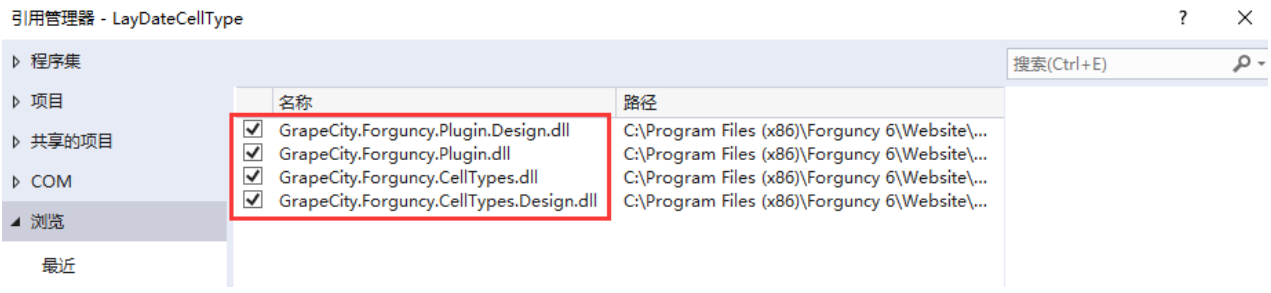
4

在解决方案资源管理器中，“引用”上右击，选择添加引用。

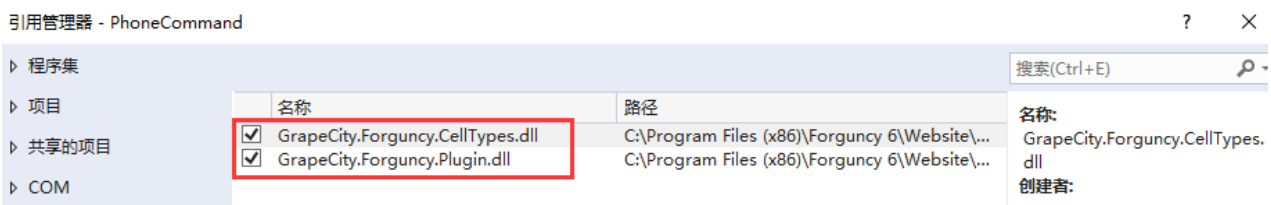
5

在右下角单击“浏览”，活字格安装文件所在的bin文件夹，如果是默认安装，则路径为"C:\Program Files (x86)\Forguncy\Website\designerbin"，找到以下文件，选中打开后，单击“确定”将其添加到解决方案资源管理器中。

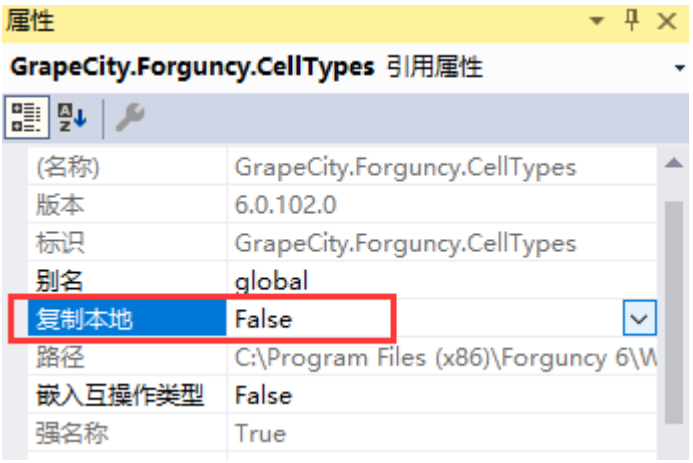
- 如果您要自定义celltype designer或者要实现ICellTypeChecker 等接口，找到GrapeCity.Forguncy.CellTypes.dll、GrapeCity.Forguncy.CellTypes.Design.dll、GrapeCity.Forguncy.Plugin.dll 、GrapeCity.Forguncy.Plugin.Design.dll 这四个文件；



- 否则只需添加GrapeCity.Forguncy.CellTypes.dll 和 GrapeCity.Forguncy.Plugin.dll 这两个文件。



并且将这些文件中的“复制本地”设置为“False”以确保最佳性能。



6

在LayDateCellType.cs文件中添加代码。

添加“默认值”和“选择模式”两个属性，设置默认的水平对齐为右对齐，实现接口IReferenceFormula，当您将单元格粘贴到另一个位置时，如果它包含单元格公式或是一个单元格范围，则默认值将会更新。

```
[Designer("LayDateCellType.LayDateCellTypeDesigner,LayDateCellType")]
[Icon("pack://application:,,,/LayDateCellType;component/Resources/Icon.p
ng")]
public class LayDateCellType : CellType, IReferenceFormula
{
    [DisplayName("")]
    [FormulaProperty]
    public object DefaultValue
    {
        get;
        set;
    }
    [DisplayName("")]
    public LayDateMode LayDateMode
    {
        get;
        set;
    }
    public override string ToString()
    {
        return "Lay";
    }

    public override DisplayBehaviour DisplayBehaviour
    {
        get
        {
            return DisplayBehaviour.KeepBorderWhenMerge;
        }
    }

    public IEnumerable<LocatedObject<object>>
GetFormulaReferObjects(LocationIndicator location)
    {
        yield return new LocatedObject<object>(this.DefaultValue,
location.AppendProperty(""));
    }
}
```

```
public class LayDateCellTypeDesigner :
CellTypeDesigner<LayDateCellType>, ISupportStyleInitialize
{
    public Dictionary<StylePropertyName, object>
GetDefaultStyleInfos(ICellInfo cellInfo)
    {
        var styles = new Dictionary<StylePropertyName, object>();
        //LayDateCellType
        styles.Add(StylePropertyName.HorizontalAlignment,
ForguncyCellHorizontalAlignment.Right);
        return styles;
    }
}
public enum LayDateMode
{
    [Description("")]
    Date,
    [Description("")]
    Time,
```

```
    [Description("&")]
    DateTime
}
```

7. 下载laydate.js文件压缩包，并将其中的theme文件夹及js文件复制到Resources文件夹下。下载地址为http://www.layui.com/laydate/。

8. 打开Resource文件夹，将laydate.js文件属性中的“复制到输出目录”规则修改为“如果较新则复制”。

9. 编辑PluginConfig.json文件，将laydate.js和theme\\default\\laydate.css文件添加到其中。

```
{
  "assembly": [
    "LayDateCellType.dll"
  ],
  "javascript": [
    "Resources/LayDateCellType.js",
    "Resources/laydate.js"
  ],
  "css": [ "Resources\\theme\\default\\laydate.css" ],
  "image": "Resources/PluginLogo.png",
  "description": "该插件支持三种输入模式：日期、时间及日期&时间，同时支持默认值设置和绑定值设置。",
  "name": "Lay日期",
  "pluginType": "cellType",
  "guid": "e62b2eaf-730a-466a-a089-e5b0d255e6e1",
  "version": "1.0.0.0",
  "dependenceVersion": "6.0.102.0"
}
```

10. 在LayDateCellType.js文件中添加代码。

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

    //CellTypeBase
    LayDateCellType.prototype.createContent = function () {
        var self = this;
        var element = this.CellElement;
        var cellTypeMetaData = element.CellType;
        var container = $("<div id='" + this.ID + "'></div>");
        var innerContainer = $('<input type="text" id=' + this.ID +
'_laydate />');
        innerContainer.css("width", element.Width);
        innerContainer.css("height", element.Height);
        innerContainer.css("box-sizing", "border-box");
        innerContainer.css("border", "0px");
        innerContainer.css("outline", "none");
        var hAlign = this.getHorizontalAlignment(element.StyleInfo);
        if (hAlign) {
            innerContainer.css("text-align", hAlign);
        }
        container.append(innerContainer);
        return container;
```

```
    }

    //
    LayDateCellType.prototype.getHorizontalAlignment = function
(styleInfo) {
        if (styleInfo) {
            if (styleInfo.HorizontalAlignment ===
Forguncy.CellHorizontalAlignment.Left) {
                return "left";
            } else if (styleInfo.HorizontalAlignment ===
Forguncy.CellHorizontalAlignment.Center) {
                return "center";
            } else if (styleInfo.HorizontalAlignment ===
Forguncy.CellHorizontalAlignment.Right) {
                return "right";
            }
        }
        return null;
    }

    //CellTypeBase
    LayDateCellType.prototype.getValueFromElement = function () {
        return this.OADateValue;
    }
    //CellTypeBase
    LayDateCellType.prototype.setValueToElement = function (element,
value) {
        if (!(value instanceof Date)) {
            if (typeof (value) === "number") {
                value = Forguncy.ConvertOADateToDate(value);
            } else {
                try {
                    value = new Date(value);
                } catch (e) {
                    value = null;
                }
            }
        }
        var info = this.getDateCellTypeTypeAndFormat();
        var type = info.type;
        var format = info.format;
        if (value == null) {
            laydate.render({
                elem: "#" + this.ID + "_laydate",
                type: type,
                format: format
            });
        } else {
            laydate.render({
                elem: "#" + this.ID + "_laydate",
                type: type,
                format: format,
                value: value
            });
        }
    }
```

```
    //CellTypeBase
    LayDateCellType.prototype.getDefaultValue = function () {
        var cellTypeMetaData = this.CellElement.CellType;
        var defaultValue = cellTypeMetaData.DefaultValue;
        return {
            Value: defaultValue
        };
    }

    //CellTypeBase
    LayDateCellType.prototype.onLoad = function () {
        var info = this.getDateCellTypeTypeAndFormat();
        var type = info.type;
        var format = info.format;
        var self = this;
        laydate.render({    //laydate.js
            elem: "#" + this.ID + "_laydate",
            type: type,
            format: format,
            done: function (value, date, endDate) {
                var newValue = Forguncy.ConvertDateToOADate(new
Date(date.year, date.month - 1, date.date, date.hours, date.minutes,
date.seconds));
                if (type === "time") {
                    newValue = newValue % 1;
                } else if (type === "date") {
                    newValue = Math.floor(newValue);
                }
                self.OADateValue = newValue;
                self.commitValue();    //
            }
        });
    }

    //
    LayDateCellType.prototype.getDateCellTypeTypeAndFormat = function ()
{
        var cellTypeMetaData = this.CellElement.CellType;
        var type = "date";
        var format = "yyyy-MM-dd";
        if (cellTypeMetaData.LayDateMode === LayDateMode.Time) {
            type = "time";
            format = "HH:mm:ss";
        } else if (cellTypeMetaData.LayDateMode ===
LayDateMode.DateTime) {
            type = "datetime";
            format = "yyyy-MM-dd HH:mm:ss";
        }
        return {
            type: type,
            format: format
        };
    }
    return LayDateCellType;
}(Forguncy.CellTypeBase));

//c#
```

```
var LayDateMode = {
    Date: 0,
    Time: 1,
    DateTime: 2
};
```

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

结束

在插件名称上右击，选择“生成”或者“重新生成”。完成后重启活字格设计器，新建的插件就会安装到活字格设计器中。

回到顶部