

帮助文档

.NET Framework 4.7.2 version 1.0

文档适配程序版本：1.0.0.0 Preview及更高

编写人：张一鸣

帮助文档

面向用户

概述

术语

权限

典型使用场景

数据填写方式

使用方法

主页构成

- ① 菜单栏
- ② 工具栏
- ③ 切换模板下拉列表
- ④ 数据采集区
- ⑤ 快捷动作区
- ⑥ 预览区
- ⑦ 预览区工具栏

工作流程

- 手动填写
- 文本解析
- 文件数据导入
- 批量打印

配置指南

全局配置

- 模板预设规则
- 打印API对比
- API 支持情况一览表
- 推荐接口顺序

模板配置

- 通用预留值配置
- 单框补全配置
- 联动补全设置

同步配置

- 权限
- 收发流程

常见问题

面向维护人员

概述

文件结构

- 一览表
- 扩展名说明
- 重要文件说明

部署

运行环境

部署方法
模板结构
概述
配置部分
案例
文本分析
原理
编写脚本
注意事项
交互性
CLI传参

面向用户

概述

欢迎使用单据生成工具，本程序可以根据模板生成样式统一的单页图文媒体，您可以将其保存为图片文件、网页或者直接将其打印。通过本教程，您可以了解以下基本操作：

- 程序的典型使用场景
- 数据填写方式
- 生成结果
- 模板设置
- 全局配置
- 同步设置

术语

文档中出现的部分术语可能造成歧义，在此先行定义：

- 模板：**由管理员先行编写好的样式结构，包含排版、预置的图文、和对应预留项的配置。**其宿主为一个html网页文件，遵循网页xml解析标准。**模板还可以加载资源**，此时模板就不止一个文件。
- 图文媒体：由于本工具主要功能围绕打印展开，一个图文媒体指的是**一个填充了对应数据的模板**，或者简单理解为“打印出的媒体”。若无特殊说明，软件中所有的“新建”指的都是新建一个图文媒体。
- 预留值：有时也称为配置项，**是指的模板中可后期插入数据的部分**，预留值有多个属性，其中最重要的是**ID**，程序将动态地查询模板中的ID并替换其内容。
- 全局设置：存储在应用配置文件中，将应用于所有模板的设置，无法分模板独立设定。
- 模板设置：存储在模板中，只应用于当前模板。

权限

因用户只使用用户界面进行操作，很多高级用法难以通过图形化实现，亦不建议普通用户修改这部分内容，包括但不限于：

- 创建模板

- 新增、删除模板配置项
- 修改模板样式
- 修改文本分析脚本

如果您有修改上述功能的需求，请联系您的系统管理员。

不过，通用配置仍对普通用户开放，您可以根据需要去定制以下内容：

- 修改全局配置：设置启动约束条件、切换打印API等。
- 修改模板内置预设：智能联想内容、智能补全内容、默认值、约束条件等。
- 发送、接收配置。

修改这些配置较为安全，不会对业务造成严重影响。

典型使用场景

1. 单页填写打印

数据可由手动、文本解析、CLI获取。

此种方式适合打印定制化的**票据、凭条、优惠券、入场券、页签、瓶签、腕带、名片、徽章等**。

2. 循环打印

打印多张相同内容的媒体，数据可由手动、文本解析、CLI获取。

当单页填写打印需要**多张副本**时，可以使用该方式打印。

3. 文件数据导入为对象（JSON）

数据存储在Excel中（仅支持.xlsx文件），将文件全部数据导入模板，得到**单张**包含多个数据的图文媒体。

此种填写方式适合打印定制化的**展示板、名单、宣传栏等**。

4. 文件数据导入为文本（批量打印）

数据存储在Excel中（仅支持.xlsx文件），每次将文件中的一行数据导入模板，得到**多张**包含单条数据的数据的图文媒体。

此种填写方式适合批量打印给定内容，通常为**团体**的凭条、入场券等。

5. 混合

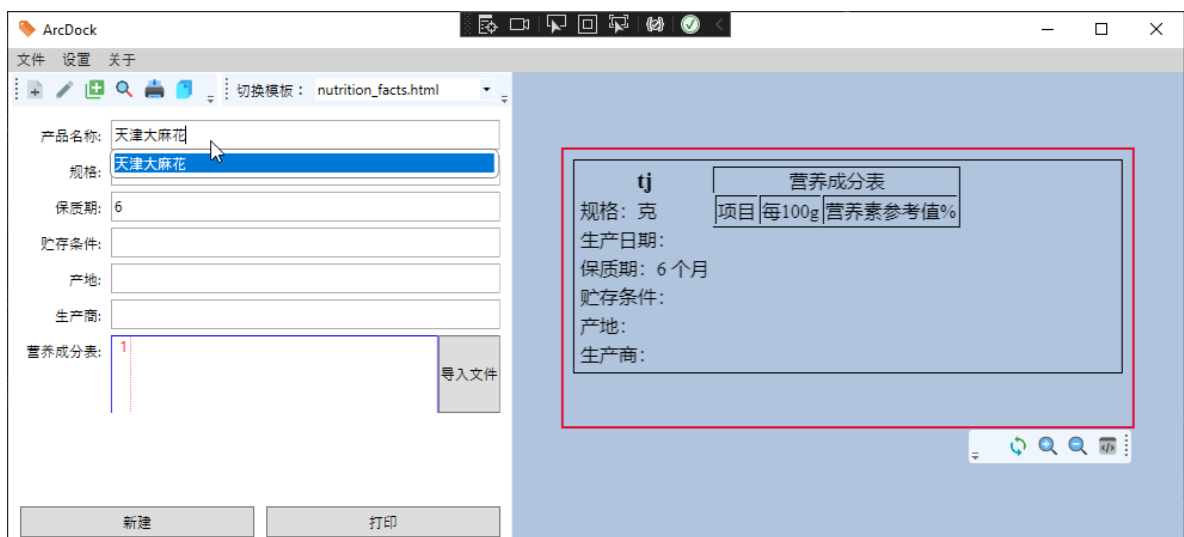
您可以将JSON保存在Excel中，通过批量打印该Excel得到上述第三条与第四条的叠加效果。

此种填写方式适合打印定制化的**数据图表、幻灯片、简单手册等**。

数据填写方式

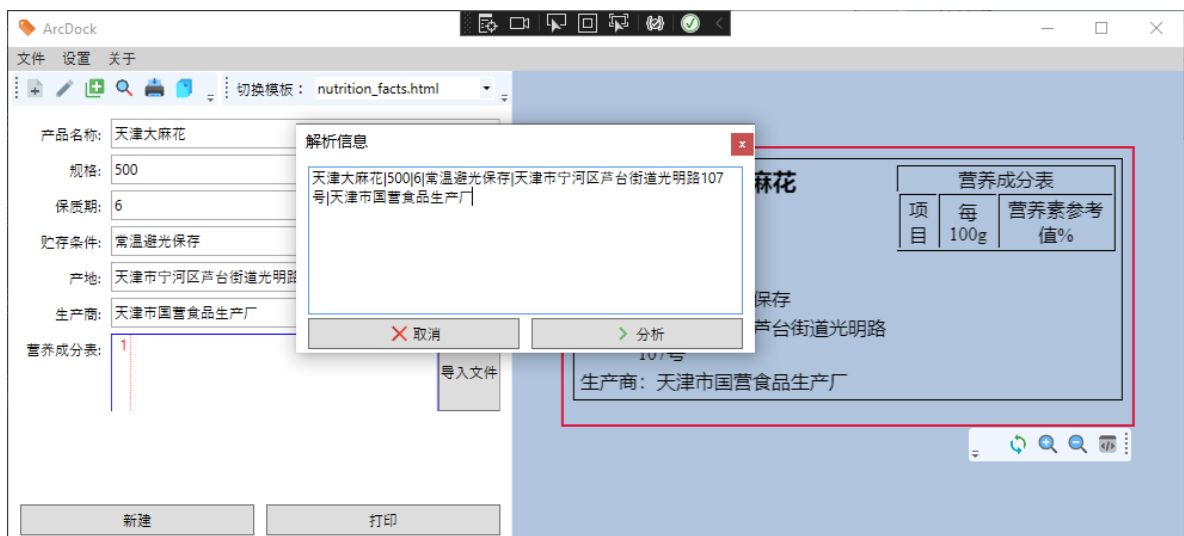
1. 手动填写

数据通过用户手动填写获取，可以使用 **智能文本框** 来联想填写内容，增加填写效率。



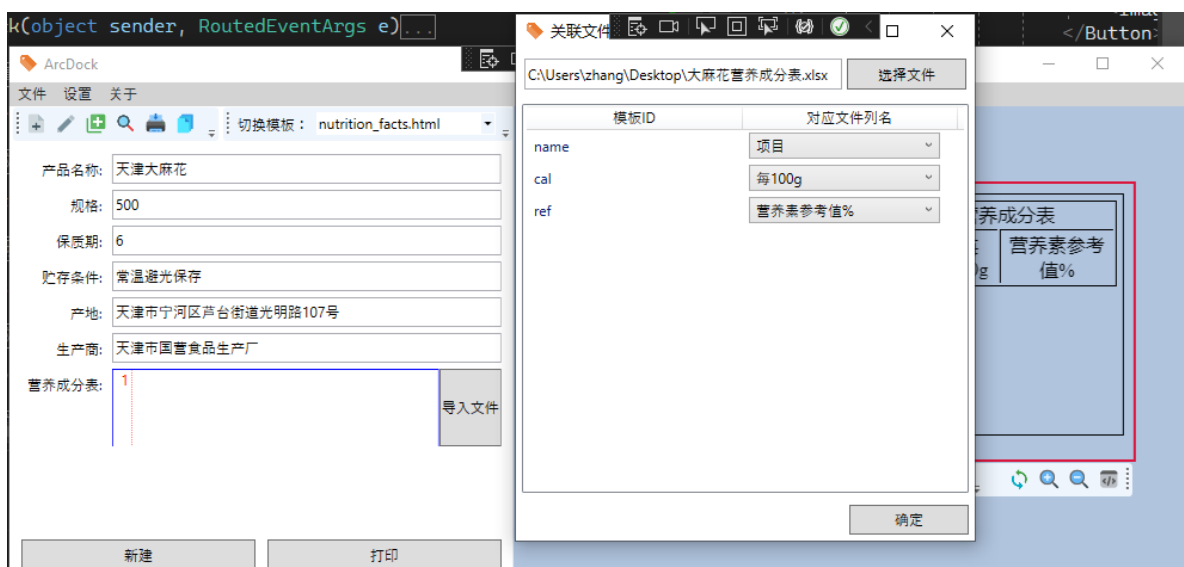
2. 文本解析

数据由文本解析获取，数据源通常来自其他软件，需要其文本可被复制。



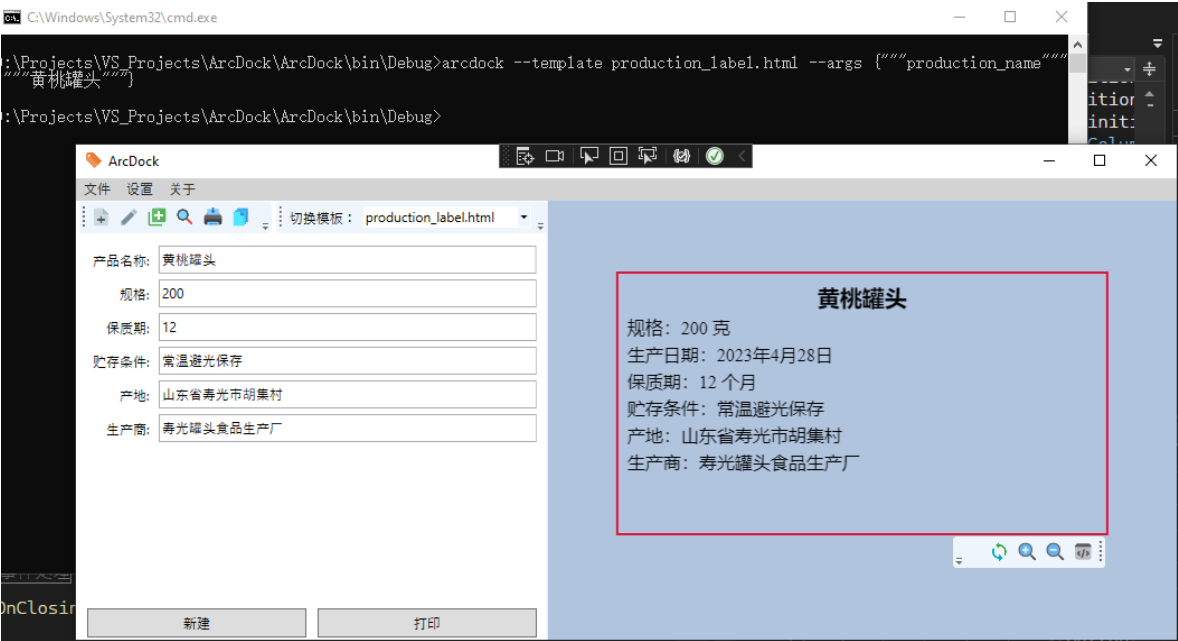
3. 来自文件

可选择将文件数据导入为对象（JSON）或文件数据导入为文本（批量打印）



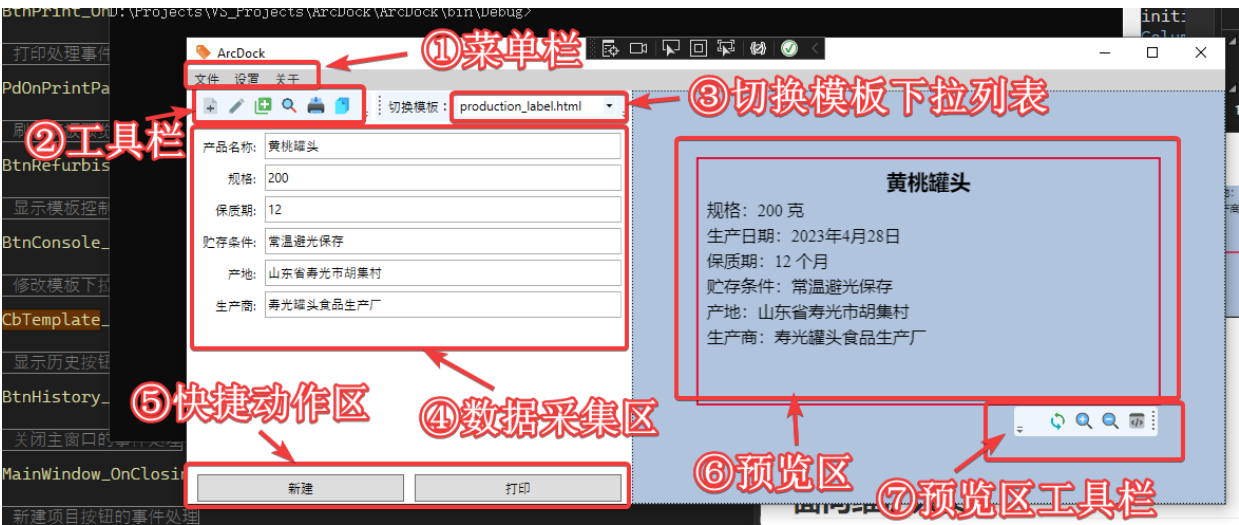
4. CLI

使用CLI界面通过命令传输数据。此方式一般用于三方软件调用。



使用方法

主页构成



① 菜单栏

菜单栏有三个独立菜单组成，分别为 文件、设置、关于，其中：

- 文件
 - 新建：新建一个图文媒体，旧媒体不会保存。
 - 导入文件：读取指定Excel并按行批量打印，打印前可以选择列名和预留值的对应关系。
 - 导出为
 - HTML网页：将当前图文媒体导出为HTML网页
 - JPG图片：将当前图文媒体导出为JPG图片也可以通过系统内置PDF打印机导出为PDF，软件内不再提供。
 - 打印

- 打印单份：打印一份图文媒体
- 打印多份：打印多份图文媒体（并非所有打印API都支持系统内多份打印，详情请参考[“打印API对比”](#)章节）
- 设置
 - 全局设置：修改全局设置
 - 模板设置：修改模板设置（并非所有模板设置都支持在系统内修改，如有高级需求请联系管理员）
 - 同步配置
 - 发送当前配置：启动Socket服务器并发送当前配置
 - 接收配置：接收其他客户端发送的配置
- 关于
 - 软件信息：软件版本、依赖包、更新说明等信息
 - 使用说明：打开说明文档

② 工具栏

目前提供的工具从左到右依次为 新建、文本分析、批量导入、查看历史、打印、批量打印



- 新建：（同 文件 → 新建）新建一个图文媒体，旧媒体不会保存。
- 文本分析：调出文本分析输入窗口。
- 批量导入：（同 文件 → 导入文件）读取指定Excel并按行批量打印，打印前可以选择列名和预留值的对应关系。
- 查看历史：调出打印历史窗口。
- 打印单份：（同 文件 → 打印 → 打印单份）打印一份图文媒体。
- 打印多份：（同 文件 → 打印 → 打印多份）打印多份图文媒体（并非所有打印API都支持系统内多份打印，详情请参考[“打印API对比”](#)章节）

③ 切换模板下拉列表

在此可切换成功解析的模板，模板文件位于程序所在目录的 `template` 目录中，应以html后缀名结尾。

④ 数据采集区

在此可以手动录入预留值数据。根据预留值配置不同，文本框有以下几种形式：

- 普通文本框：

产地:

由标题和内容组成的普通文本框，不可手动换行。

- 多行文本框

生产商:

内容
内容
内容

普通文本框的可换行版本，但是是否真正可在模板中体现为多行显示还需要模板样式控制。

- 智能文本框

产品名称:
规格:

智能文本框可在用户输入内容拼音头或者一部分内容时给出备选项，提高输入效率。有两种版本：

- 单框文本补全：只是在用户输入时给出补全提示，辅助用户完成本预留值的填写。
- 多框联动补全：在用户输入的内容命中提示、或者选择给出的提示时，根据内容也改变其他文本框的值。

两种智能文本框提示内容和行为都可以在模板配置中修改，详情请见[“模板配置”](#)章节。

- JSON对象

营养成分表:

1 {"name":["能量","蛋白质","脂肪","碳水化合物"],
2 "cal":["2404kJ","8.2g","26.0g","54.9g"],
3 "ref":["0.29","0.14","0.6","0.18"]}

导入文件

由标题、内容、导入文件按钮组成，用户可以选择点击 导入文件 按钮选择Excel文件生成JSON对象，也可以手动编写JSON对象。

⑤ 快捷动作区

提供两个常用快捷操作：新建 和 打印（单份），功能与 文件 → 新建 和 文件 → 打印 → 打印单份 相同。

⑥ 预览区

预览数据填充的效果，一般在切换模板、数据填充后进行刷新，若未进行刷新，可以使用预览区工具栏的 强制刷新 触发强制刷新。

根据打印API不同，预览效果可能与实际打印效果不同，详情参见[“打印API对比”](#)章节。

⑦ 预览区工具栏

预览区工具栏从左到右依次为 强制刷新、放大、缩小、控制台。



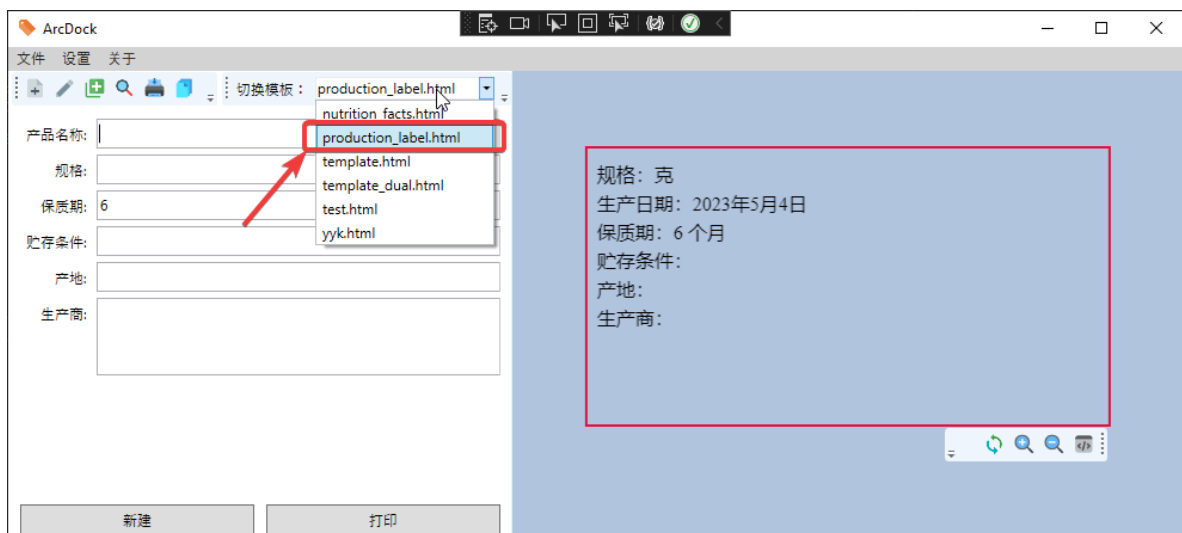
- 强制刷新：强制刷新预览区的页面。
- 放大：放大当前预览区页面，**仅放大预览效果，打印时使用模板配置。**
- 缩小：缩小当前预览区页面，**仅缩小预览效果，打印时使用模板配置。**
- 控制台：打开预览区页面的控制台，一般由管理员调试时使用。

工作流程

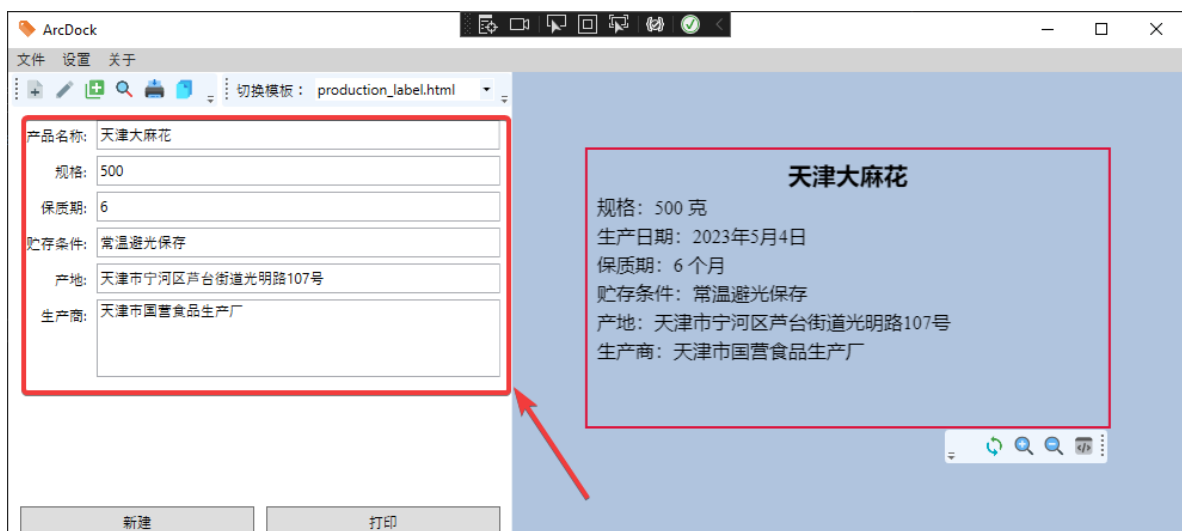
以下使用“食品标签”作为示例，展示了几种典型使用场景。

手动填写

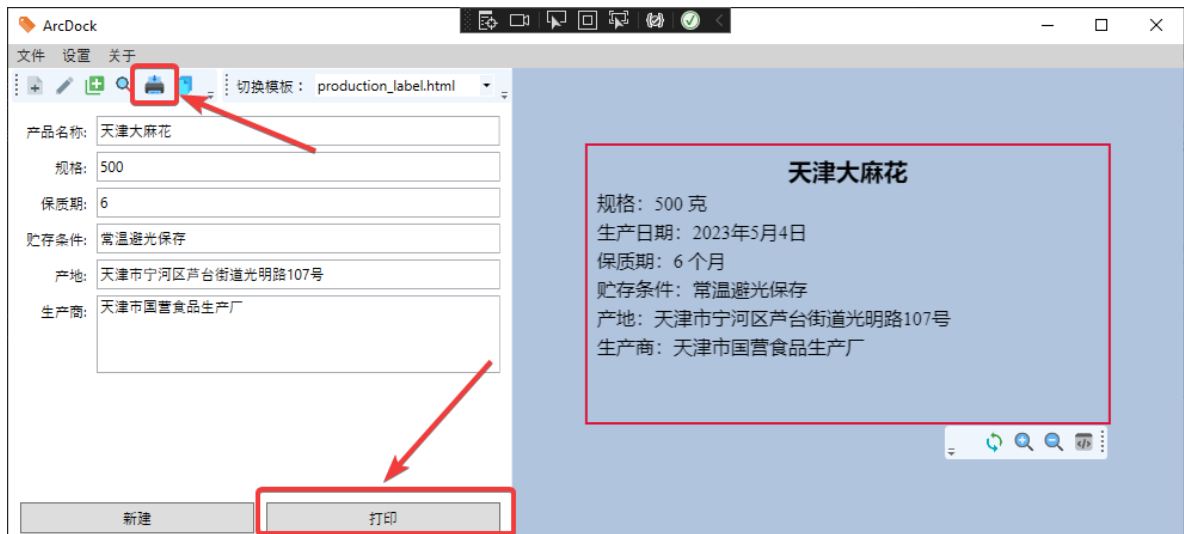
1. 选择正确的模板



2. 在左侧数据采集区输入数据

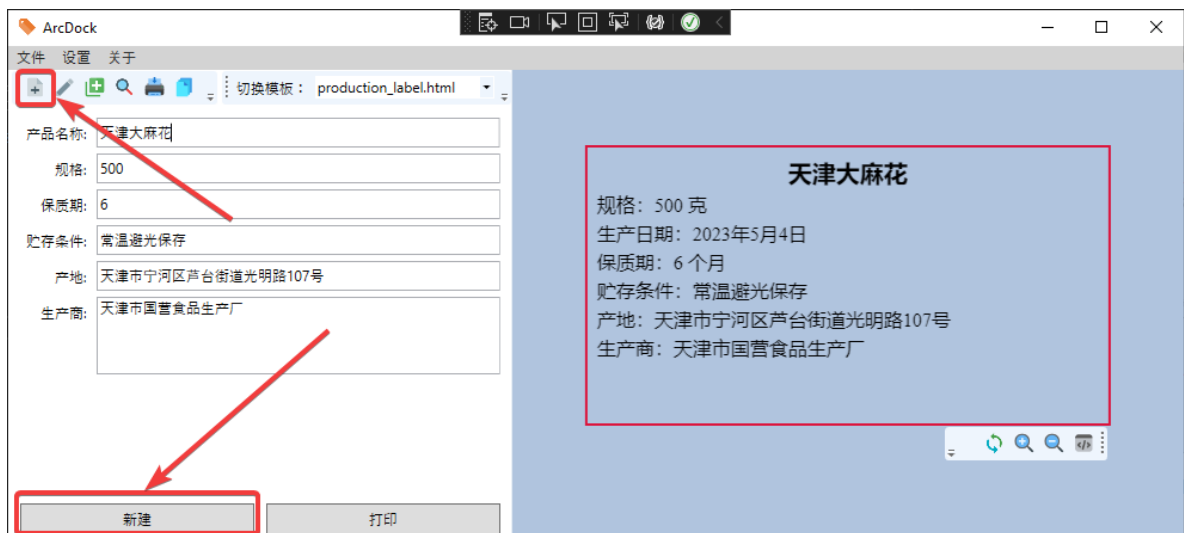


3. 在预览区中确认排版无误后，点击**工具栏**中的 打印 按钮或**快捷动作区**的 打印 按钮即可打印当前图文媒体。



4. 打印完毕后，点击**工具栏**中的 新建 或者**快捷动作区**的 新建 即可清空填入数据。

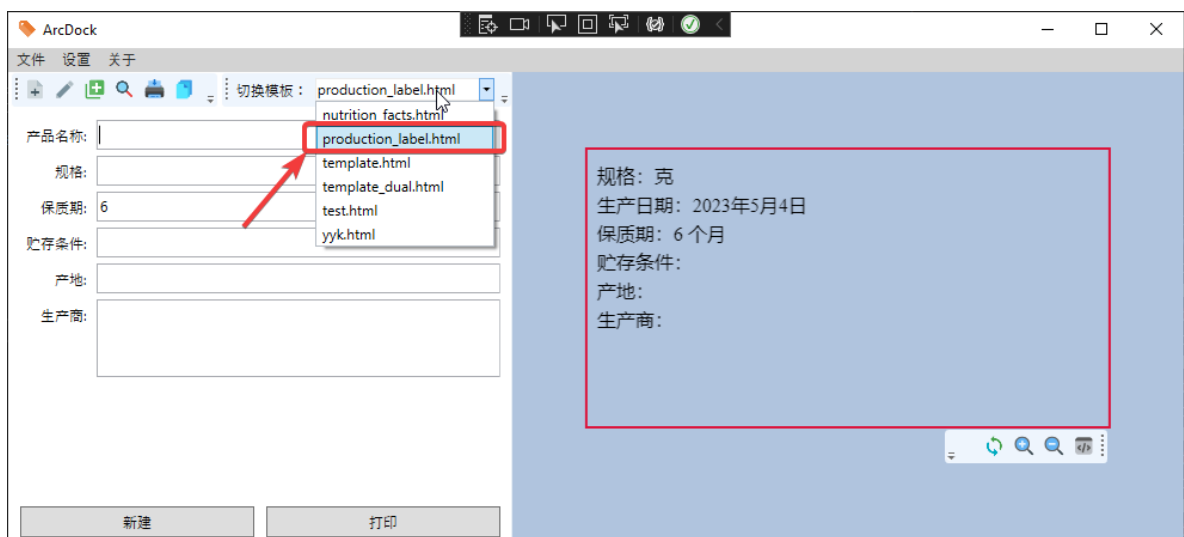
注意：因为工具栏中的按钮较为密集，容易误操作清空内容，故工具栏中的新建功能会请求用户确认后才清空内容。



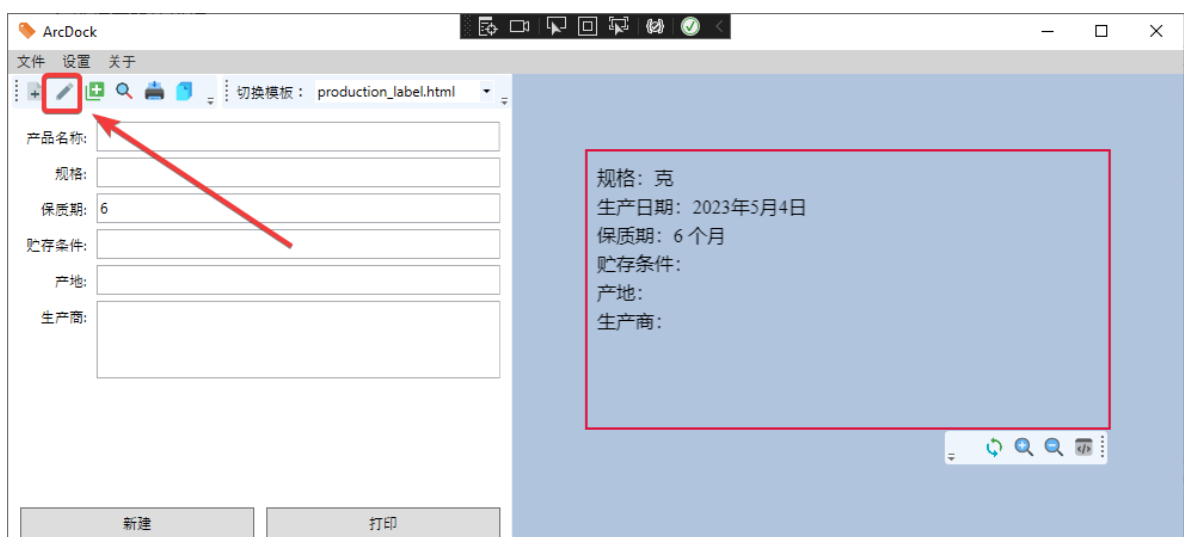
文本解析

注意：文本解析功能需要管理员事先进行配置，如果目前文本解析无法达到预期效果，请联系管理员重新配置。

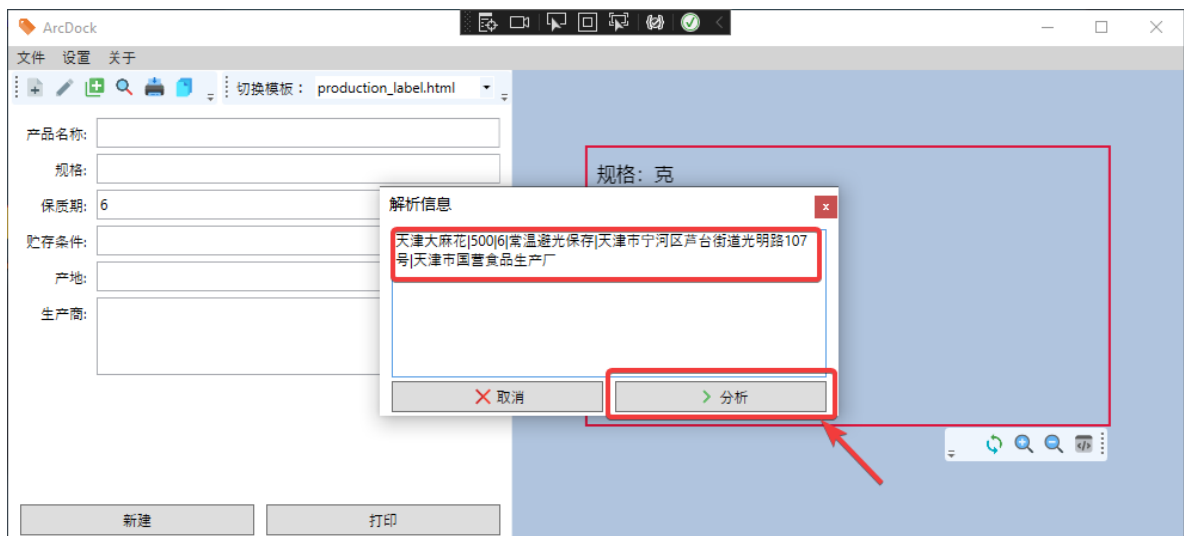
1. 选择正确的模板



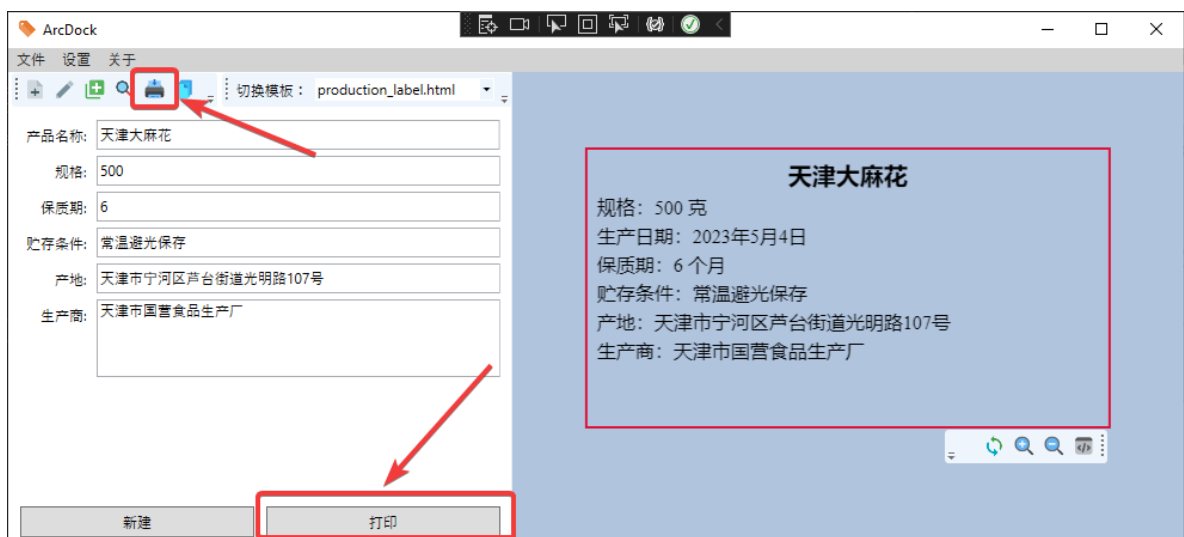
2. 点击**工具栏**中的 文本分析 按钮



3. 在弹出的窗口中粘贴来自其他系统的文本，之后点击按钮“分析”，即可填充模板数据

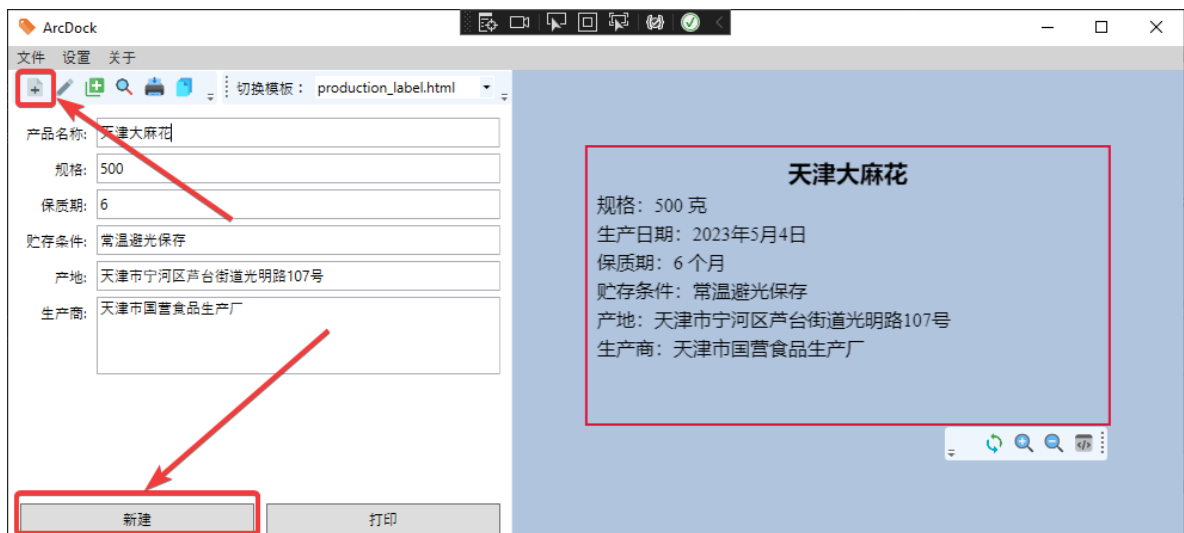


4. 在预览区中确认排版无误后，点击**工具栏**中的 打印 按钮或**快捷动作区**的 打印 按钮即可打印当前图文媒体。



5. 打印完毕后，点击**工具栏**中的 **新建** 或者**快捷动作区**的 **新建** 即可清空填入数据。

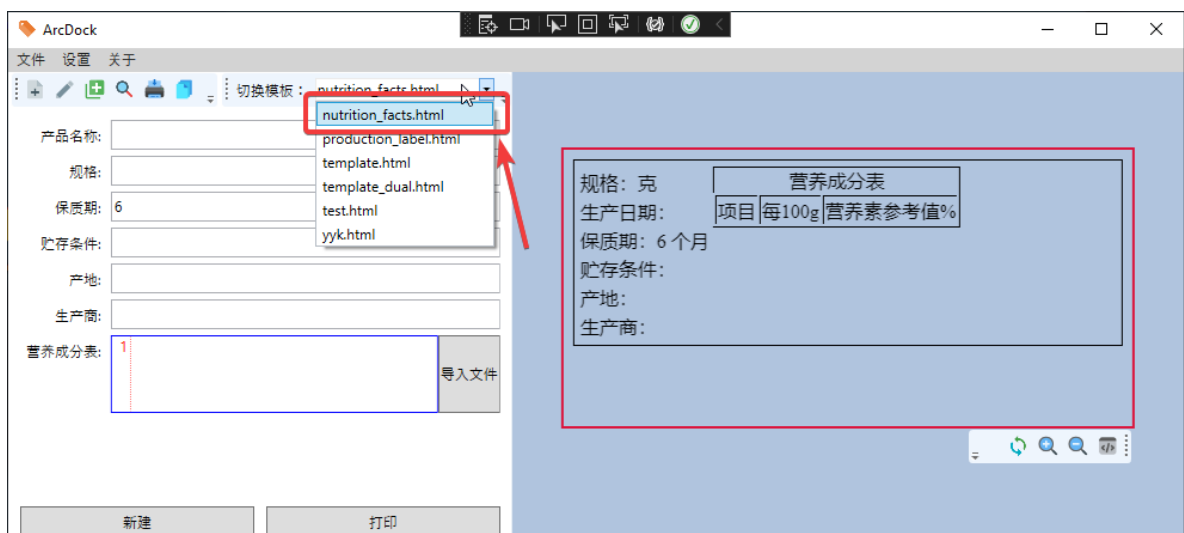
注意：因为工具栏中的按钮较为密集，容易误操作清空内容，故工具栏中的新建功能会请求用户确认后才清空内容。



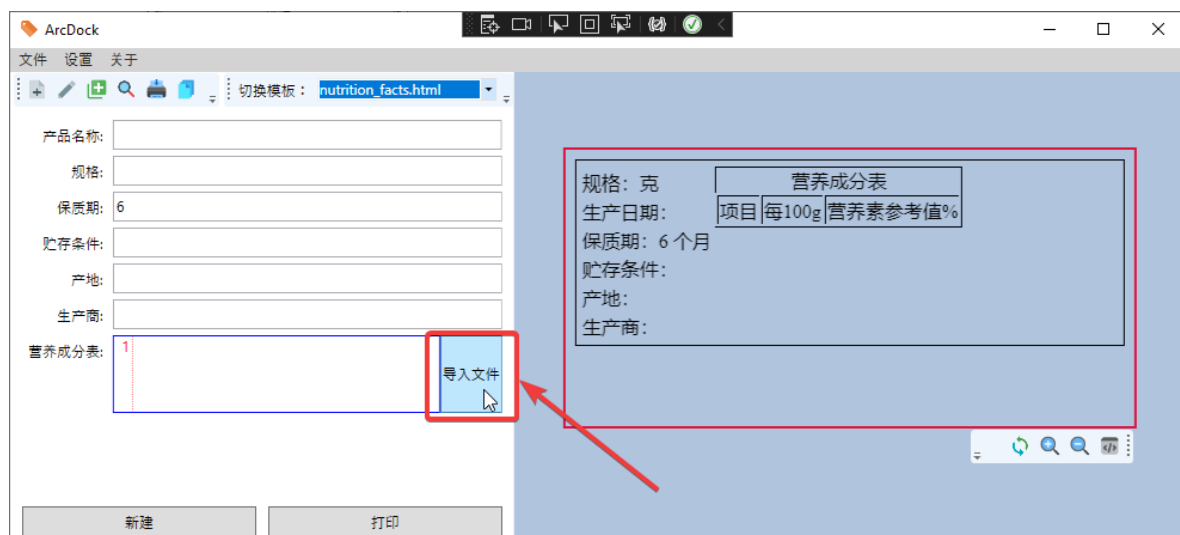
文件数据导入

注意：并非所有的模板都支持文件数据导入，如需定制，请联系管理员。

1. 选择正确的模板



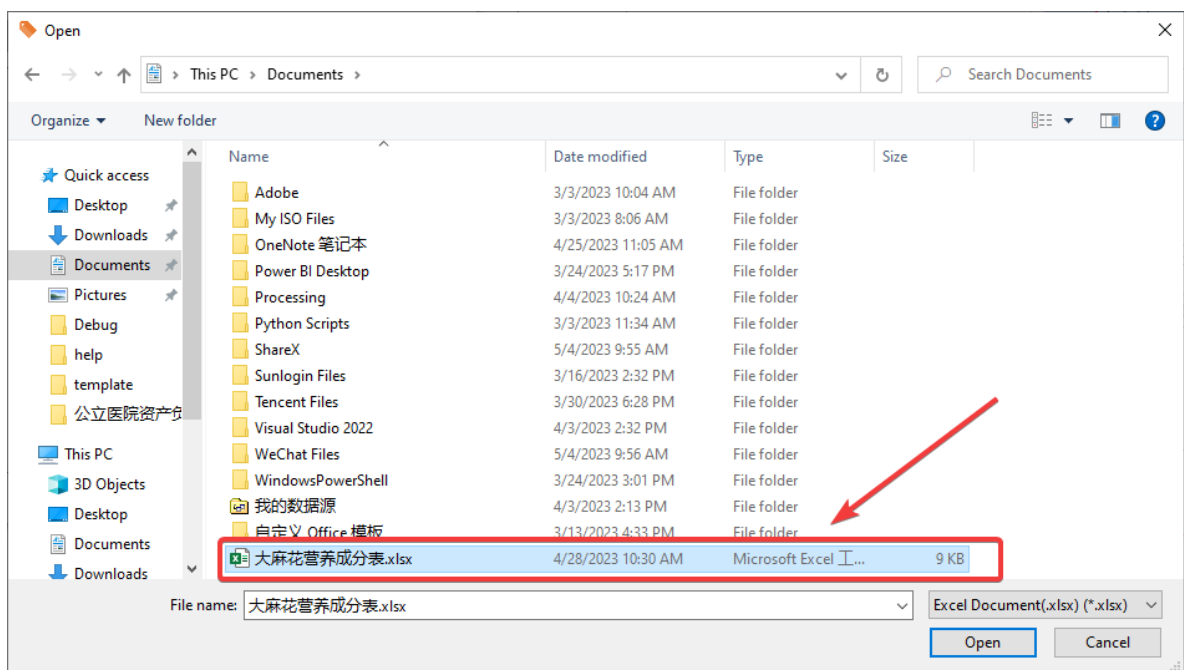
2. 在支持数据导入的预留值右侧点击“导入文件”



3. 在弹出窗口中点击“选择文件”



4. 选择需要导入的文件，目前仅支持导入xlsx格式的文件。

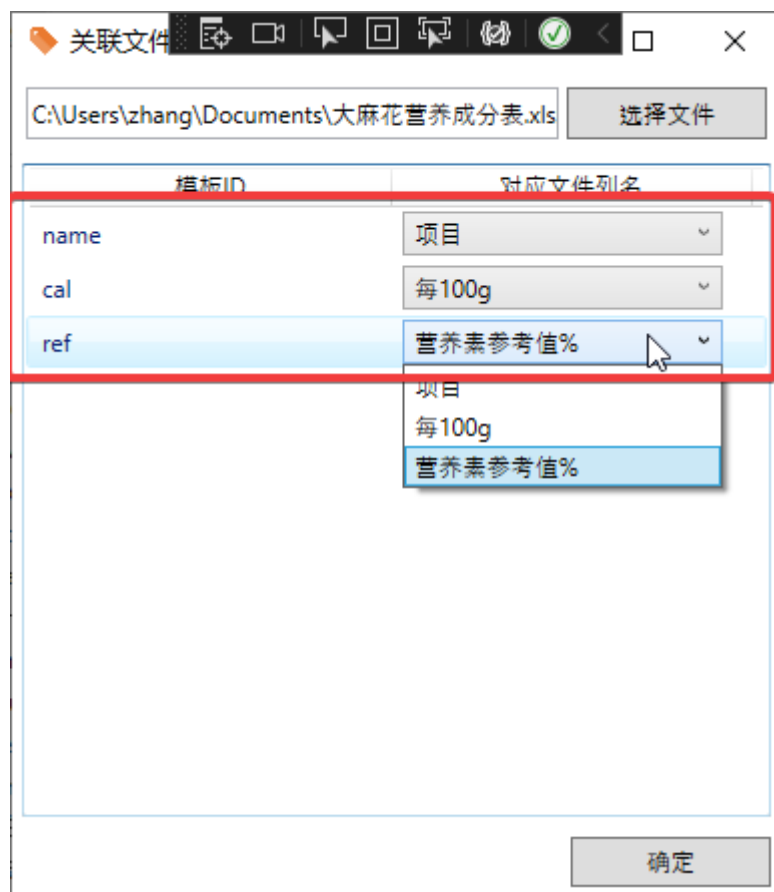


在该示例中，Excel文件“大麻花营养成分表.xlsx”的内容如下：

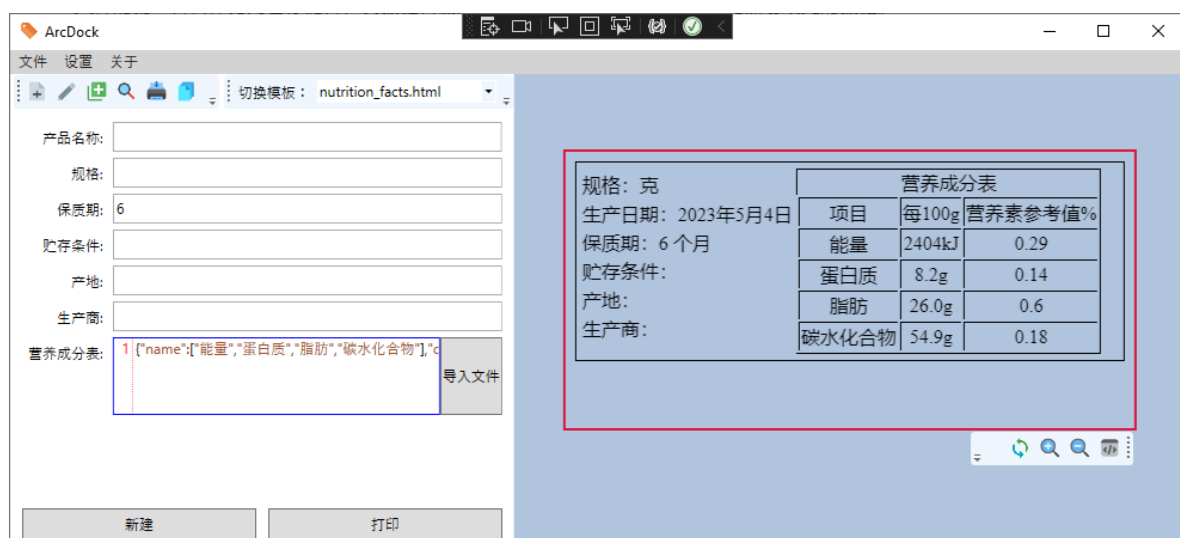
	A	B	C
1	项目	每100g	营养素参考值%
2	能量	2404kJ	29%
3	蛋白质	8.2g	14%
4	脂肪	26.0g	60%
5	碳水化合物	54.9g	18%
6			

文件的第一行将作为列名存储行，其数据将被解析为列名，请不要在第一行存储需要填充的数据。

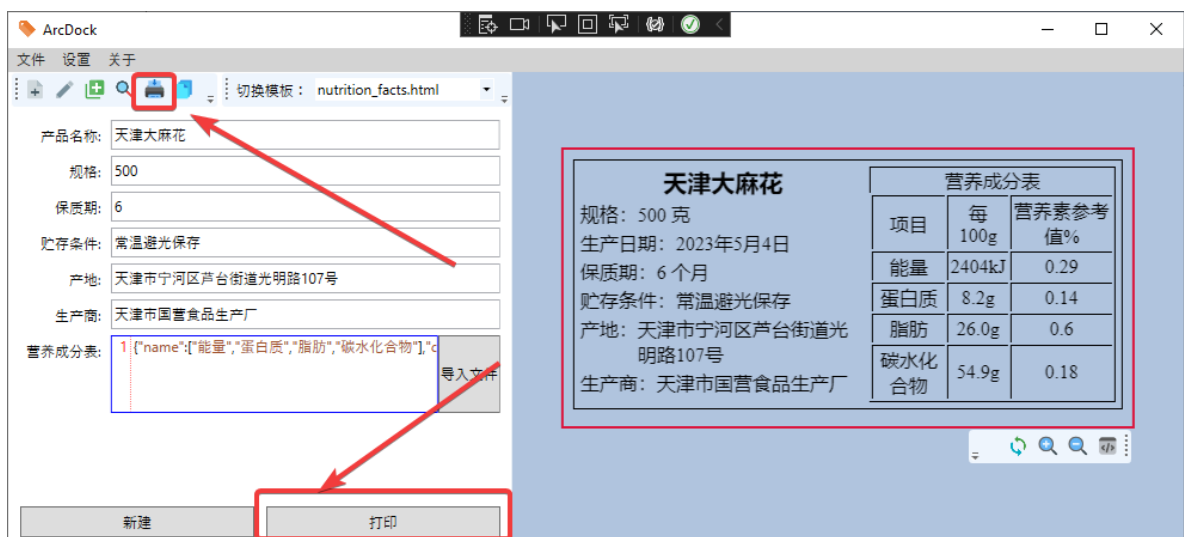
5. 确定导入的参数与文件中列名的对应关系，**默认将顺序对应**，若对应不正确可以点击下拉菜单手动指定。



6. 确认无误后点击右下角“确定”按钮，文件数据将被转换为JSON字符串填充入数据，根据模板内置规则渲染排版。

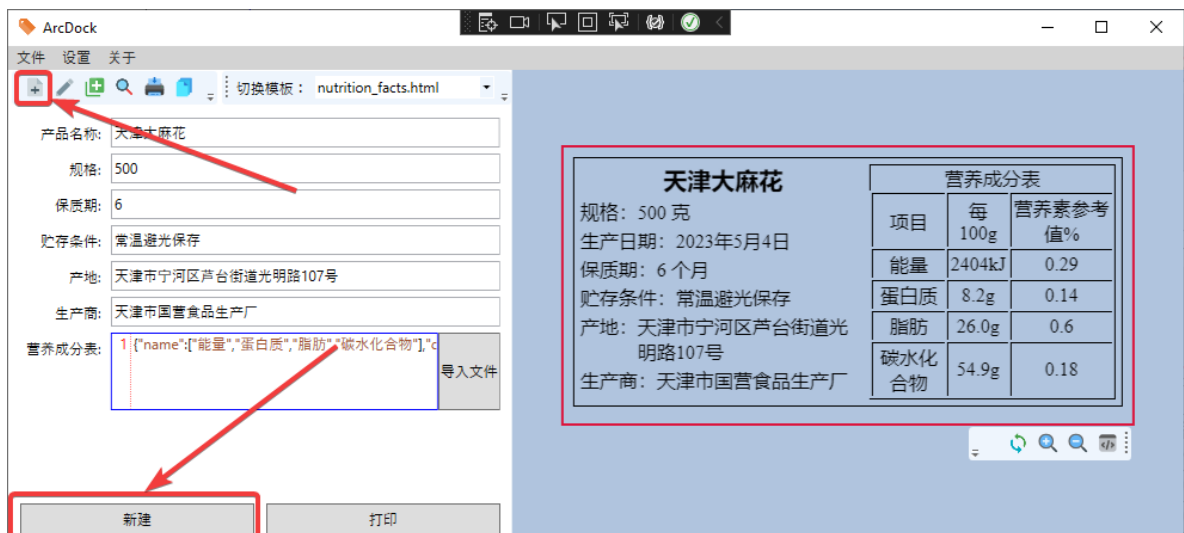


7. 填充剩余内容填充完毕后，在预览区中确认排版无误，点击**工具栏**中的 打印 按钮或**快捷动作区**的 打印 按钮即可打印当前图文媒体。



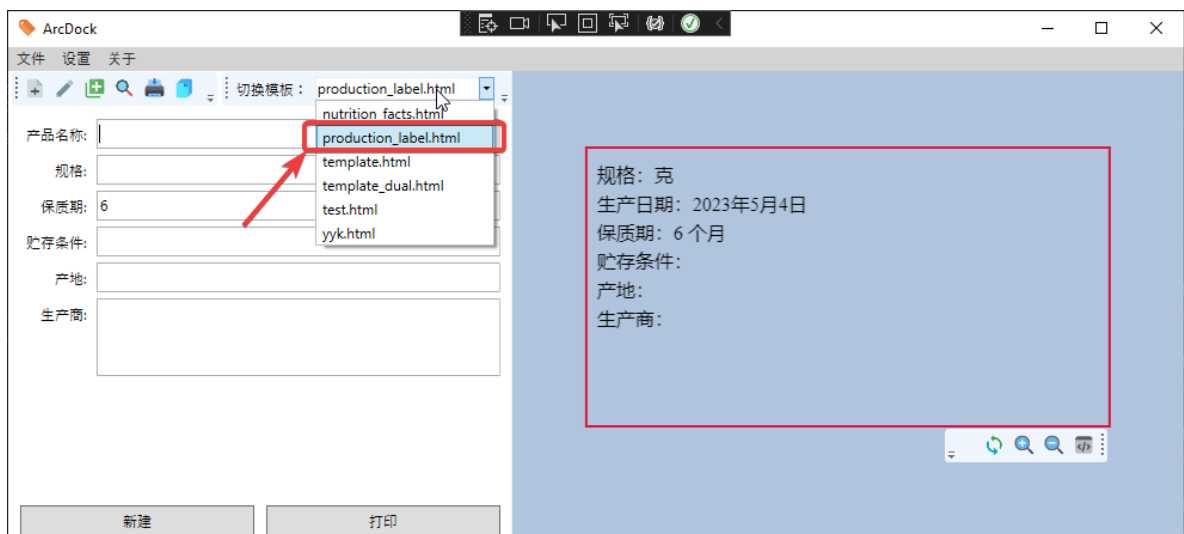
8. 打印完毕后, 点击**工具栏**中的 **新建** 或者**快捷动作区**的 **新建** 即可清空填入数据。

注意: 因为工具栏中的按钮较为密集, 容易误操作清空内容, 故工具栏中的新建功能会请求用户确认后才清空内容。

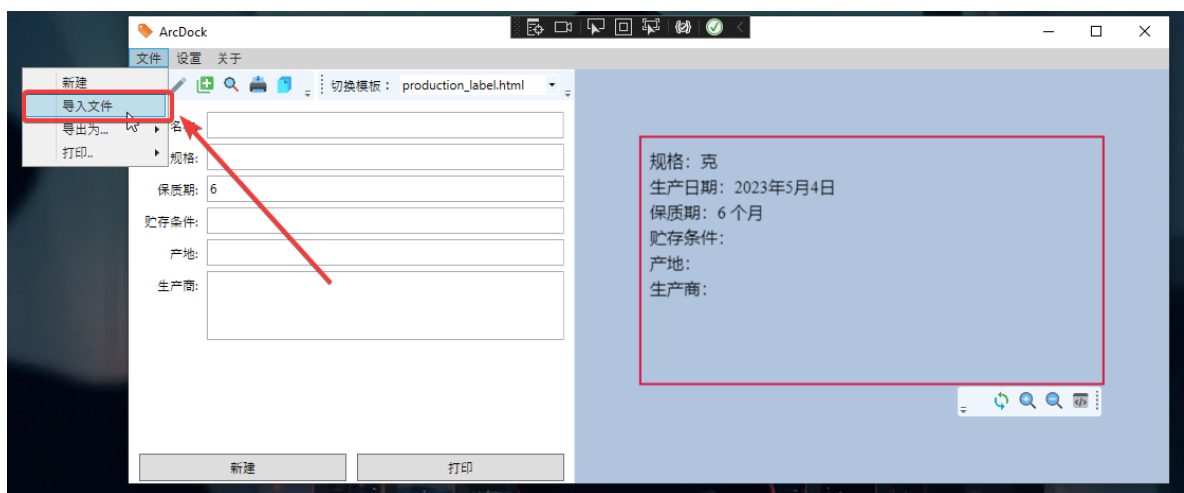


批量打印

1. 选择正确的模板



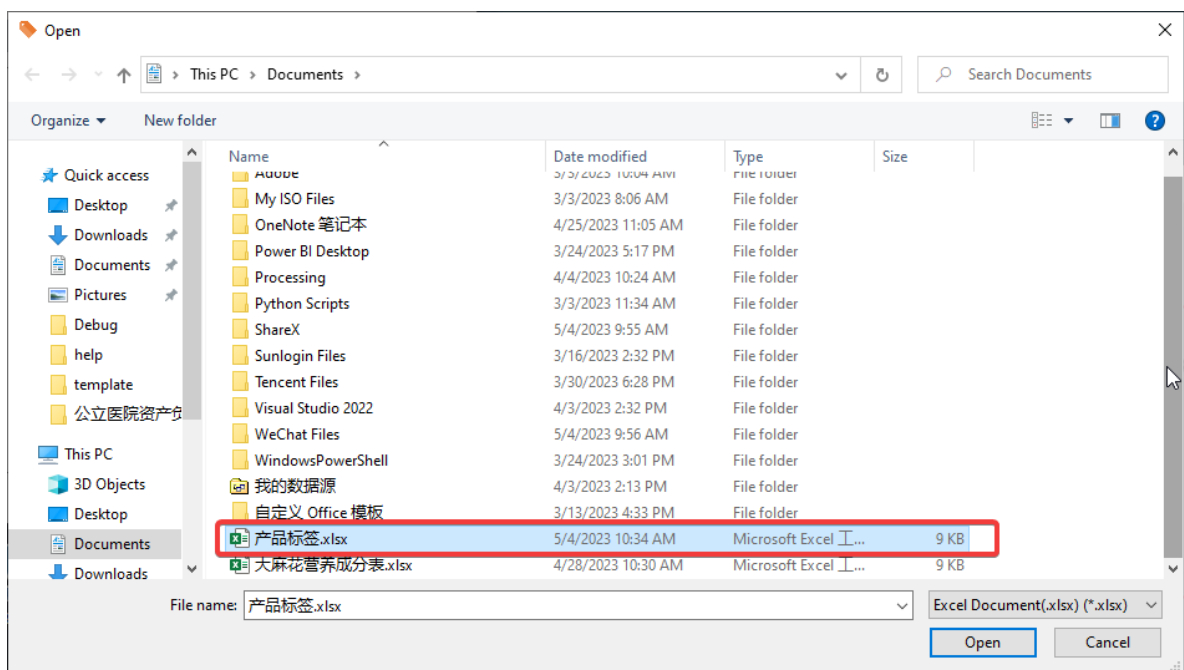
2. 在菜单栏选择 文件 → 导入文件



3. 在弹出窗口中点击“选择文件”



4. 选择需要导入的文件，目前仅支持导入xlsx格式的文件。

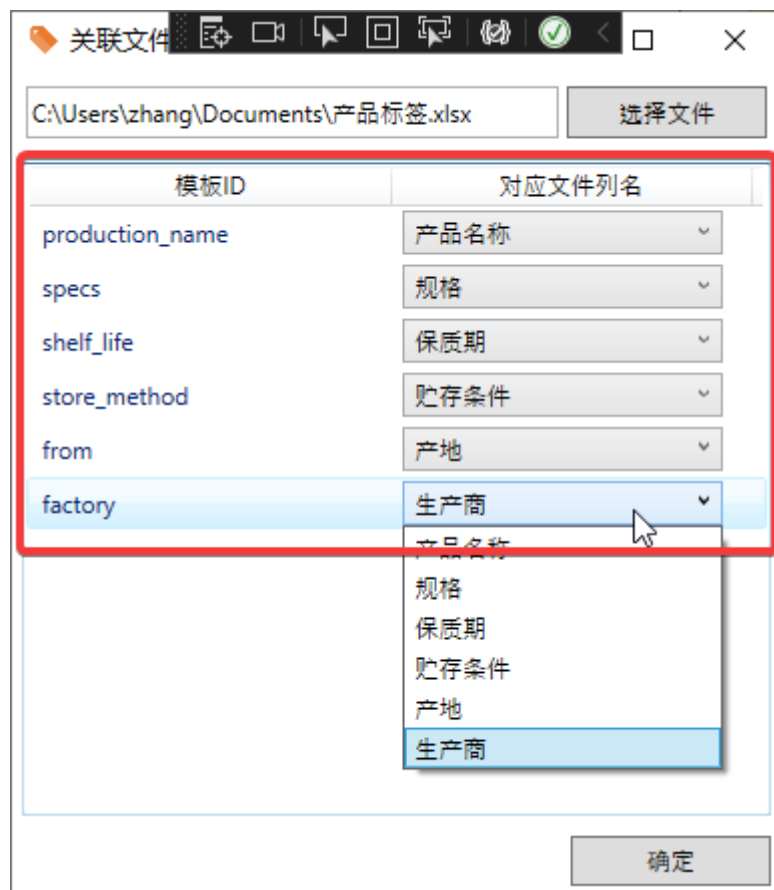


在该示例中，Excel文件“产品标签.xlsx”的内容如下：

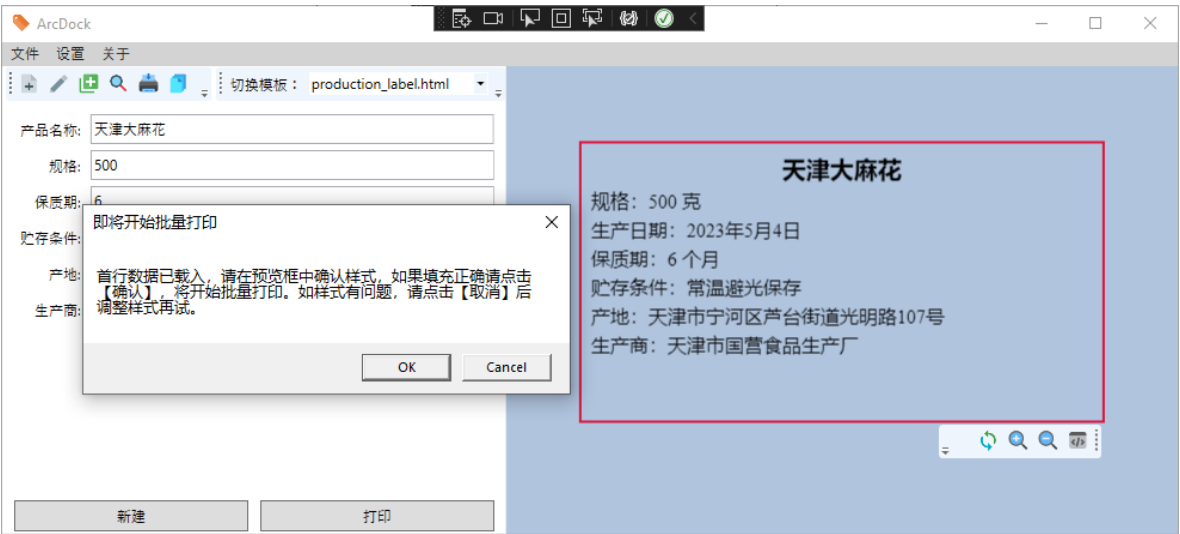
	A	B	C	D	E	F
1	产品名称	规格	保质期	贮存条件	产地	生产商
2	天津大麻花	500	6	常温避光保存	天津市宁河区芦台街道光明路107号	天津市国营食品生产厂
3	黄桃罐头	200	12	常温避光保存	山东省寿光市胡集村	寿光罐头食品生产厂
4	辣海带	120	3	包装开启前常	江苏省盐城市盐东镇建军大街220号	江苏广叶海产品加工厂

文件的第一行将作为列名存储行，其数据将被解析为列名，请不要在第一行存储需要填充的数据。

5. 确定导入的参数与文件中列名的对应关系，默认将顺序对应，若对应不正确可以点击下拉菜单手动指定。



6. 点击右下角“确定”，程序将读取文件除标题行外的第一行数据填充入模板，用户在预览框预览效果无误后点击“确认”按钮，系统将按行批量填充模板并打印。



配置指南

全局配置

模板预设规则

模板预设规则是存储在模板中对于各预留值的控制规则，匹配方式为**正则表达式**，只有当**匹配通过时才允许进行打印**。但是，有时规则较旧并没有适配新场景，就无法进行打印，**为了处理这种情况，可以在全局配置中临时禁用模板预设规则，此时任何数据都可打印。**

模板预设规则也可以通过模板配置进行修改，详情请见[“模板配置”](#)章节。

打印API对比

由于各打印接口对于打印功能的支持情况不同，有时需要根据情况更改打印API，以下为各打印API的对比：

- PrintDocument API
原理：将目前的视口截图后直接打印
优势：排版效果与预览效果相同，支持静默打印
劣势：系统原生接口打印照片时文本不清晰，打印效果一般
- CEF Print API
原理：调用CEF内置的print()函数
优势：排版效果与预览效果相同
劣势：不支持静默打印，每次点击打印都需要选择打印机
- C-Lodop API
原理：在CEF中注入CLodop接口打印函数
优势：东华系统内置了C-Lodop 打印接口，可以直接调用

劣势：没安装东华系统的电脑需要手动安装C-Lodop接口，并且接口使用IE渲染打印结果，打印效果可能与预览效果不同

• PDFtoPrinter API

原理：先使用CEF内置的方法将视口保存为PDF文件，再使用PDFtoPrinter 库的方法打印PDF

优势：排版效果与预览效果相同，支持静默打印

劣势：转储PDF文件可能比较慢

• Spire.Pdf API

原理：先使用CEF内置的方法将视口保存为PDF文件，再使用Spire.Pdf 库的方法打印PDF

优势：排版效果与预览效果相同，支持静默打印

劣势：转储PDF文件可能比较慢，Spire.Pdf是商业库，免费版不能解析超过10页的PDF文件

API 支持情况一览表

API	打印效果	静默打印	预览效果一致	指定打印机	多页打印	打印纸张规格单位
PrintDocument API	一般	√	√	√	√	Pixel (Width/Height)
CEF Print API	最好	×	√	×	×	Pixel (Html Meta Viewport)
C-Lodop API	最好	√	×	×	×	Pixel (Html Meta Viewport)
PDFtoPrinter API	好	√	√	√	√	CentiMetre (PrintWidth/PrintHeight)
Spire.Pdf API	好	√	√	√	√	CentiMetre (PrintWidth/PrintHeight)

推荐接口顺序

经验上，应首先满足**打印效果**。在打印效果可观的前提下关注**是否与预览效果一致**，以防打印错版严重（可以通过浏览器兼容性检测预防错版）。静默打印、指定打印机、多页打印为可选功能，需要的情况下可优先满足，故可推断出推荐使用的接口顺序为：

PDFtoPrinter API > CEF Print API > Spire.Pdf API > C-Lodop API > PrintDocument API

模板配置

通用预留值配置

通用预留值配置包括**预留值ID**、**友好名称**、**文本框类型**、**自动填充类型**（当文本框类型为 **智能文本框** 时）、**默认值**、**约束条件**。除预留值ID、友好名称不支持直接修改外，其他通用预留值配置均支持在模板设置界面中修改。

- 预留值ID（**不允许修改**）

最重要的预留值配置，与模板样式中的预留值ID相同，当预留值内容被修改时，程序会根据ID动态修改模板。

- 友好名称（**不允许修改**）

预留值ID的友好名称。

- 文本框类型

目前包括 普通文本框、多行文本框、智能文本框、JSON对象，详情见[④ 数据采集区](#)章节。

- 自动填充类型

仅当文本框类型为 智能文本框 时可用，目前包括 无自动补全、单框文本补全、多框联动补全。详情见[④ 数据采集区](#)章节。

- 默认值

模板加载时默认填充的数据

- 约束条件

正则表达式，当全局配置的**应用模板规则**保持开启状态，且填写数据不能匹配正则表达式时，禁止继续打印操作并提示错误。

常用正则限制：

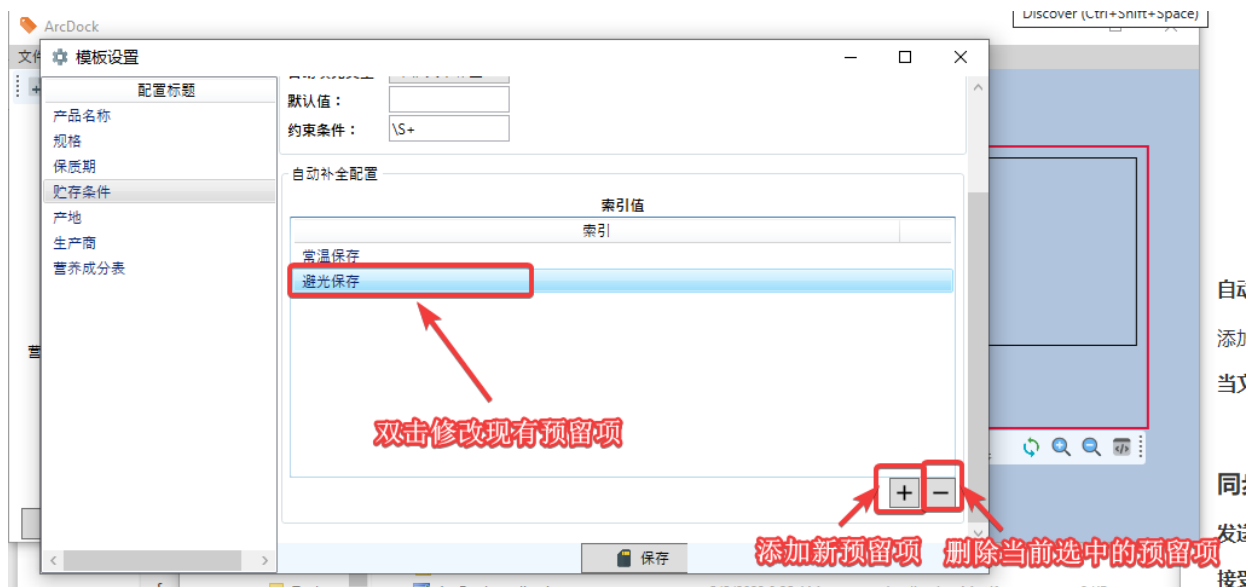
- 全部字符：\S+
- 仅数字：[0-9]
- 仅英文字符：[a-zA-Z]
- 仅中文：[\u4e00-\u9fa5]
- 仅英文和数字：[a-zA-Z0-9]

单框补全配置

单框补全指的是每次选中联想内容只会修改预留值自身的内容，不会修改其他预留值的内容。

仅当文本框类型为 智能文本框，自动填充类型为 单文本框补全 时才可进行此配置。

- 新增：点击右下角 + 新增预留项
- 删除：点击右下角 - 删除当前选中的预留项
- 修改：双击修改现有预留项

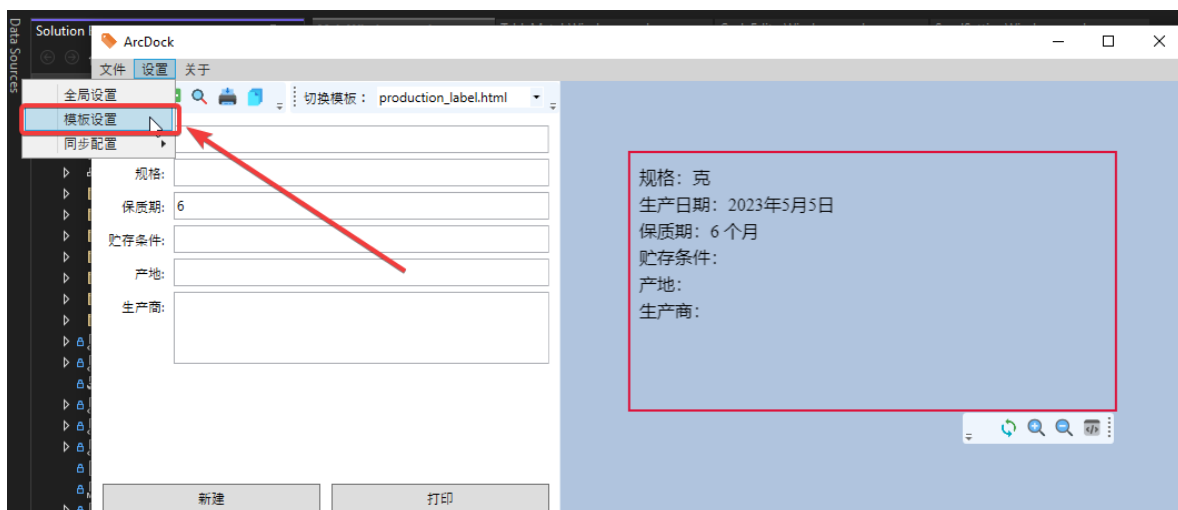


示例：智能提示贮存条件

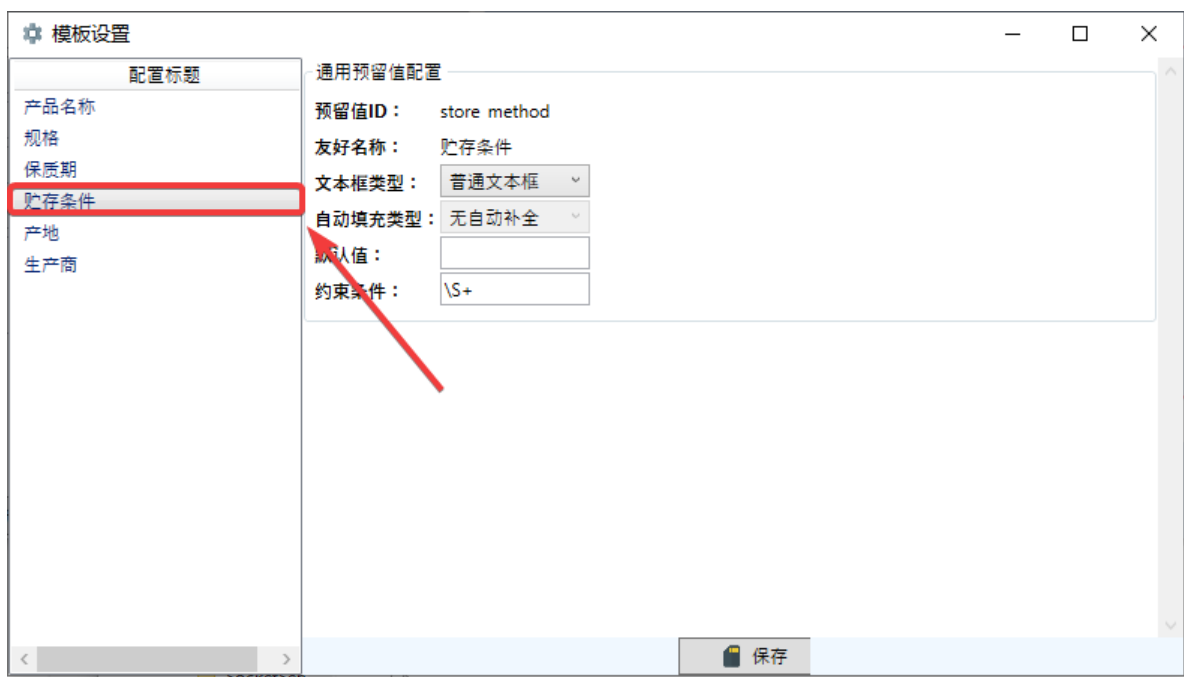
目标：在模板production_label.html中，修改贮存条件文本框类型，使之能够联想“常温保存”、“避光保存”两项。

1. 进入模板配置

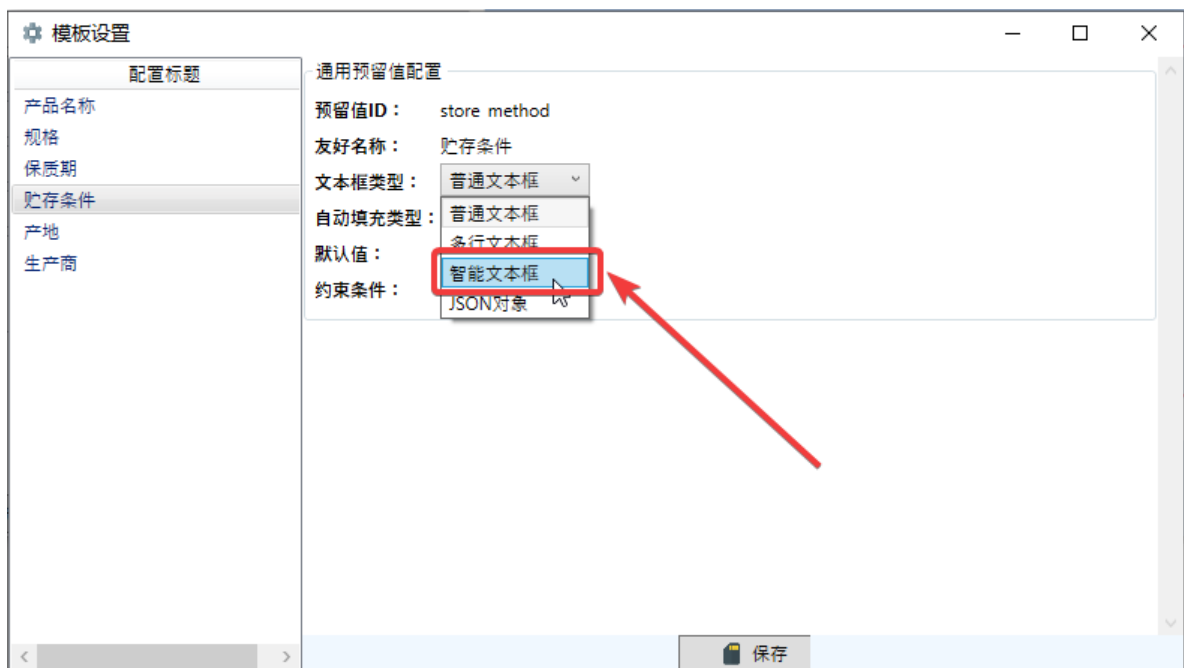
选择菜单栏 设置 → 模板设置 即可进入模板设置



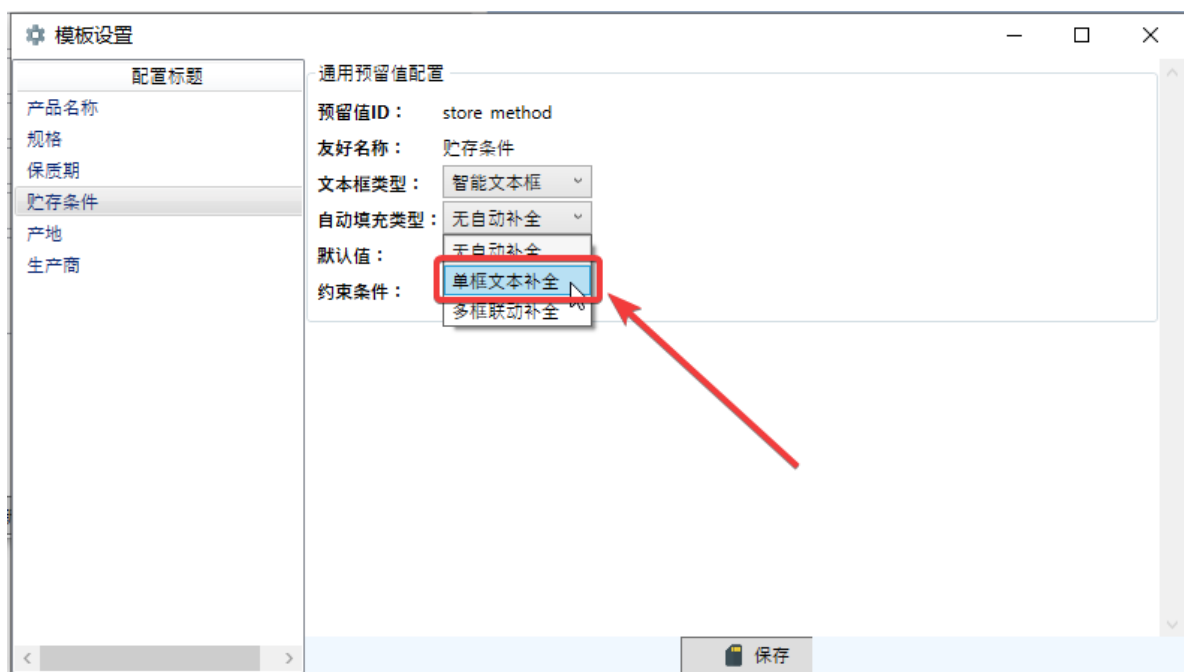
2. 在模板配置界面中左侧“配置标题”选中“贮存条件”预留值



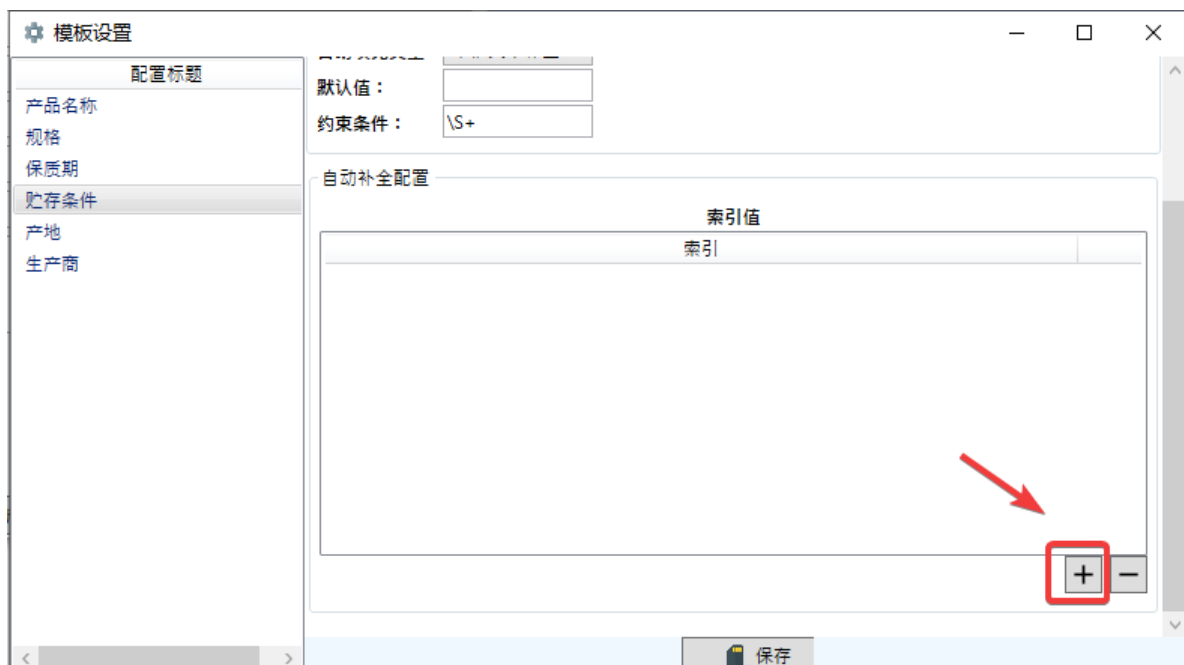
3. 将文本框类型修改为 智能文本框



4. 将自动填充类型修改为 单框文本补全



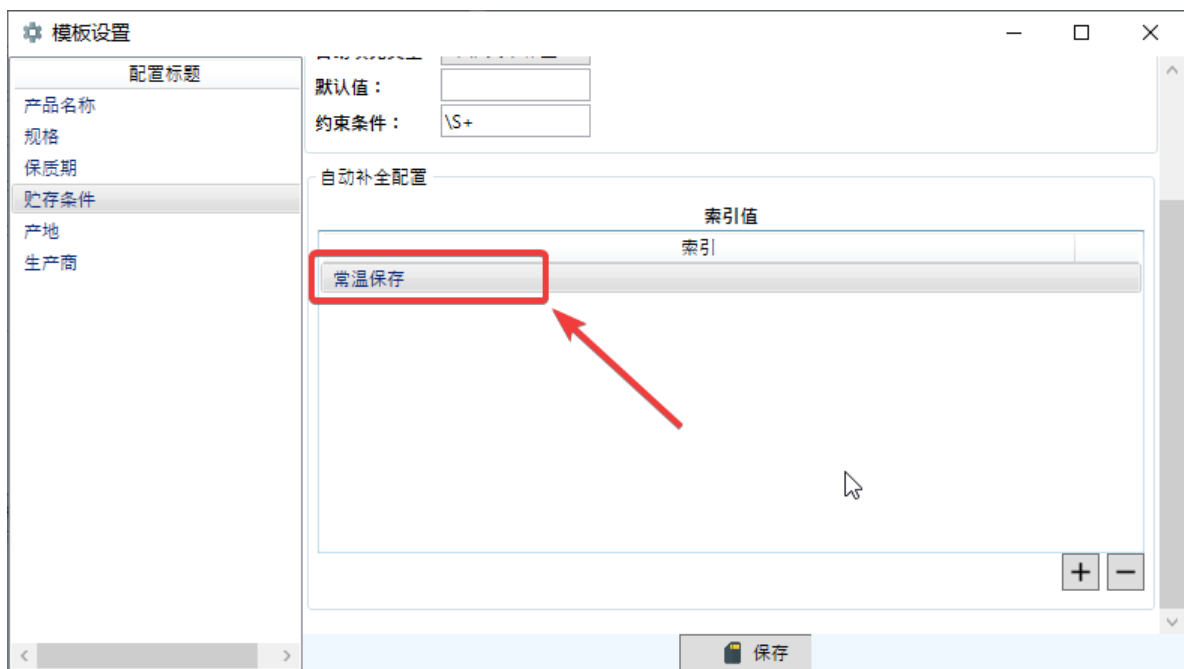
5. 在自动补全配置列表右下方点击 + 新增预置值



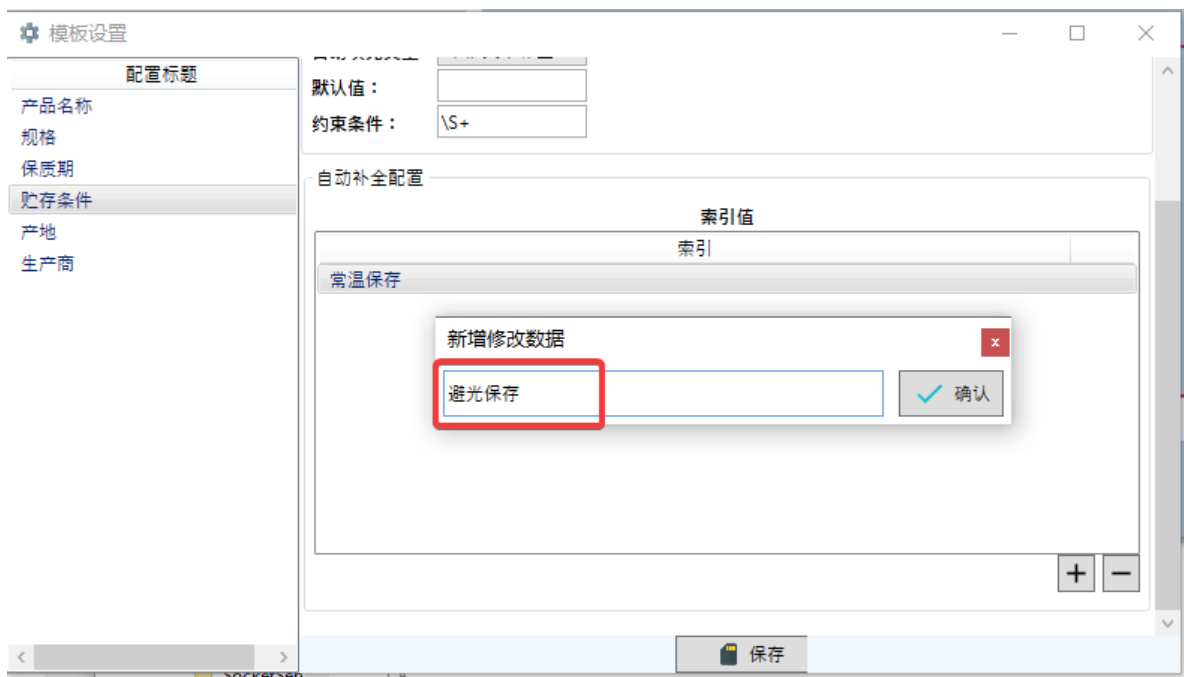
6. 在弹出输入框中输入“常温保存”，然后点击 确认



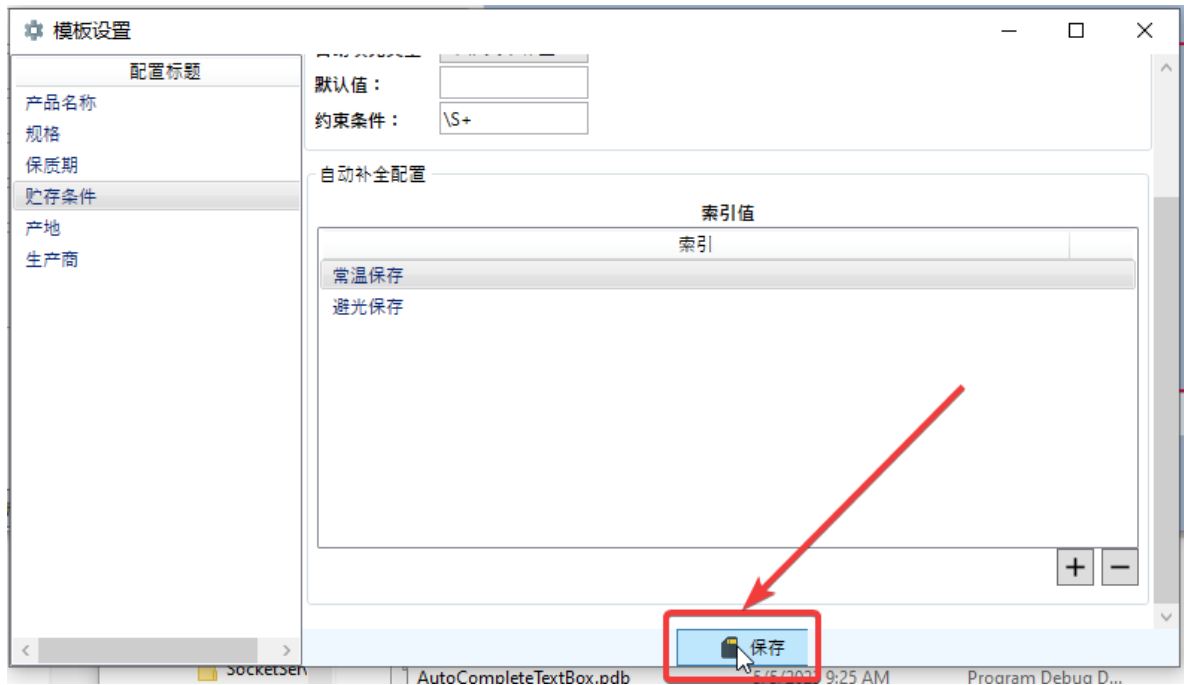
7. 此时“常温保存”备选项已经添加



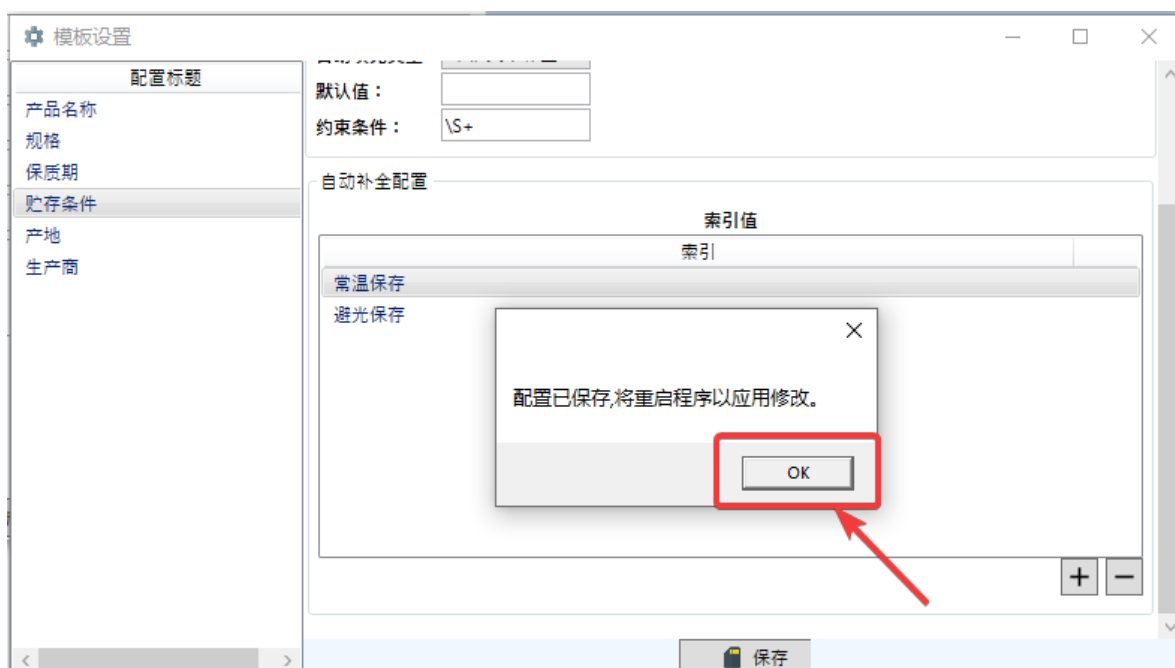
8. 使用同样的方式添加“避光保存”



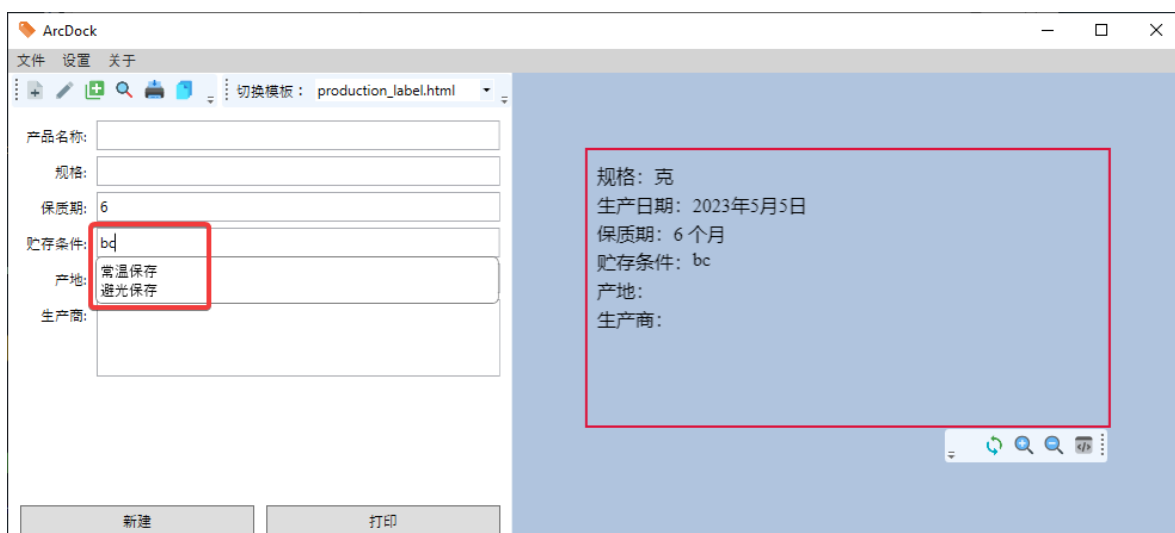
9. 添加完毕后，点击下方 保存



10. 提示重启应用，点击确定重启，若未唤起应用可以手动再打开



11. 程序启动后，在 贮存条件 中输入“保存”的拼音头“bc”，看到联想内容已成功添加。



联动补全设置

联动补全指的是每次选中联想内容不止修改预留值自身的内容，也可以修改其他预留值的内容。

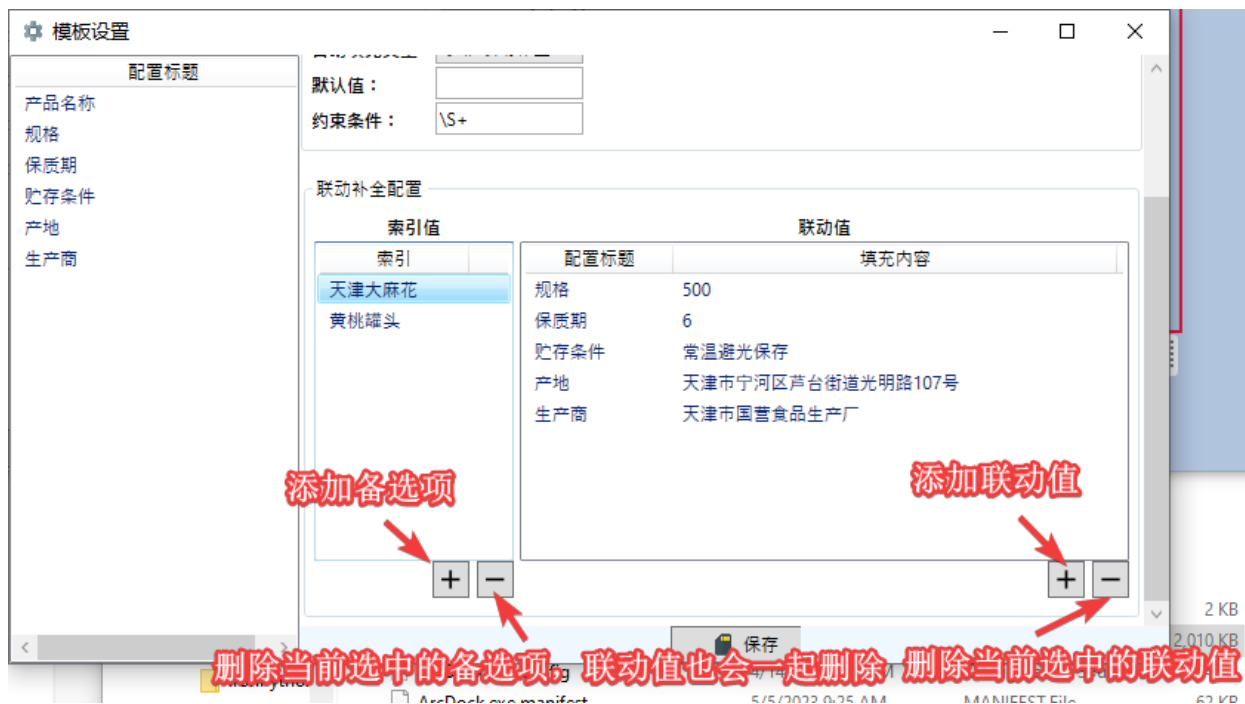
仅当文本框类型为 智能文本框，自动填充类型为 多框联动补全 时才可进行此配置。

联动补全设置由左右两个列表组成，左侧为索引值（备选项）列表，右侧为联动值列表。

索引值（备选项）指的是用户输入数据、拼音头联想的内容，**与单框补全功能相同。**

联动值指的是**当前索引值被用户选中后，其他预留项要修改的数据**，即“联动”内容。

与单框补全相似，在对应列表右下方点击 +、- 可以添加、删除选中内容，**双击内容可以编辑。**

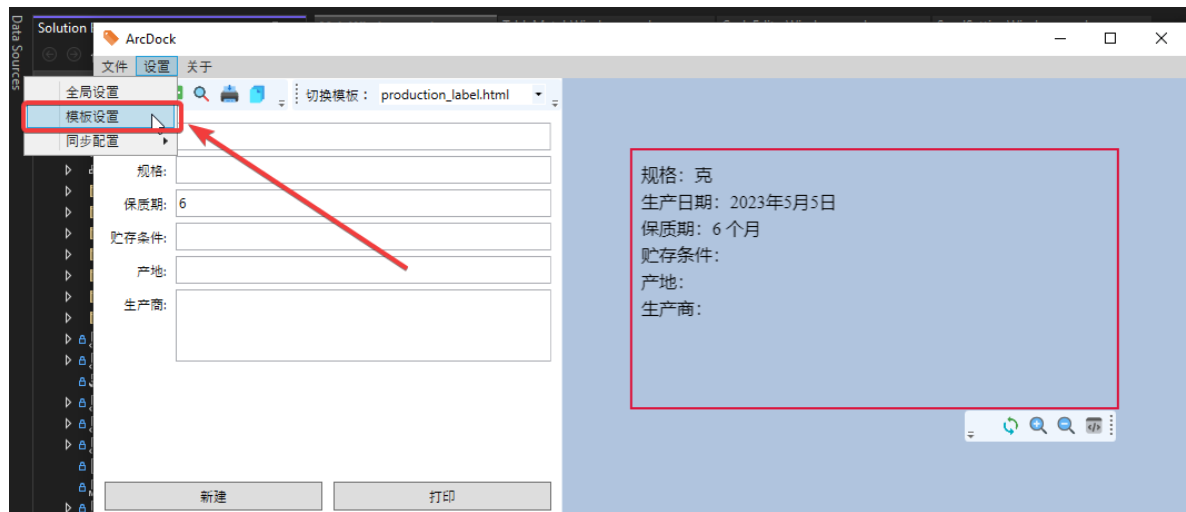


示例：添加产品“辣海带”

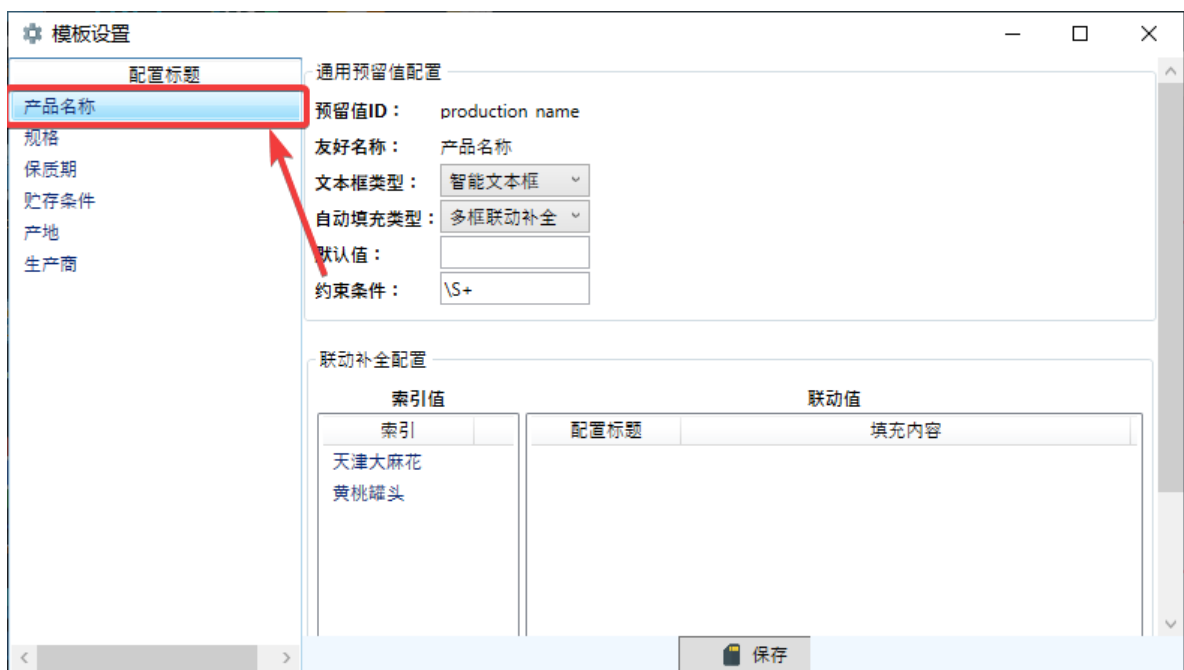
目标：用户可以通过输入拼音头快速录入产品“辣海带”，当辣海带被用户选中后，将规格填充为“120”，保质期填充为“3”，贮存条件填充为“包装开启前常温避光保存，包装开启后请冷藏保存并于7日内食用完毕”，产地填充为“江苏省盐城市盐东镇建军大街220号”，生产商填充为“江苏广叶海产品加工厂”。

1. 进入模板配置

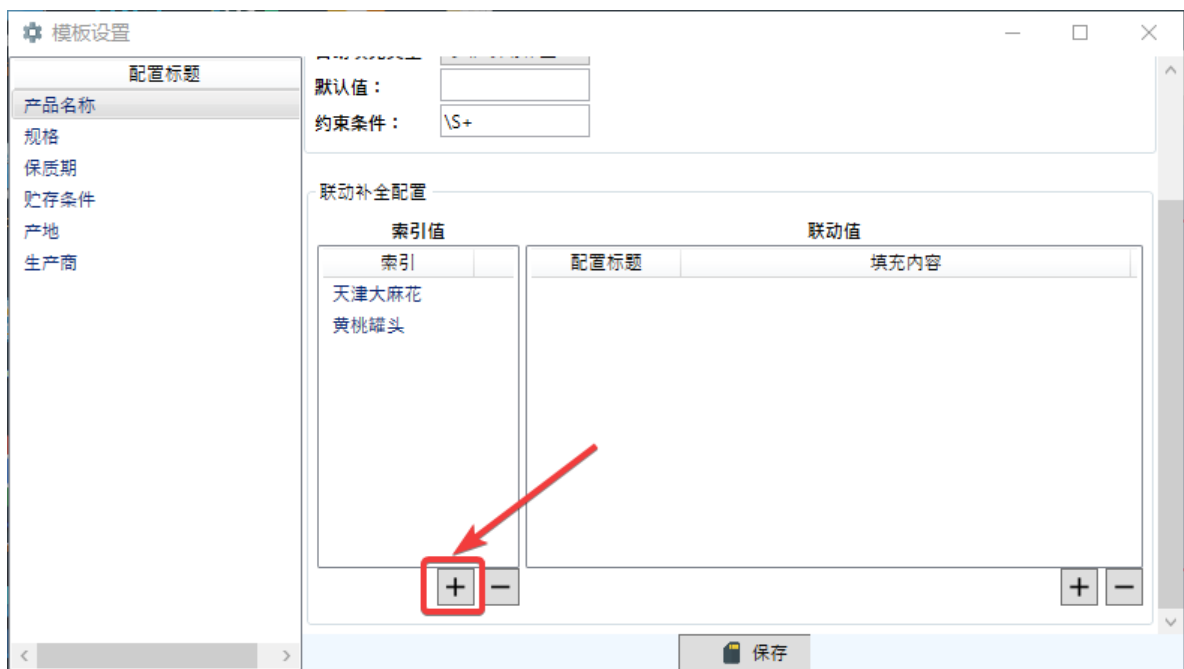
选择菜单栏 设置 → 模板设置 即可进入模板设置



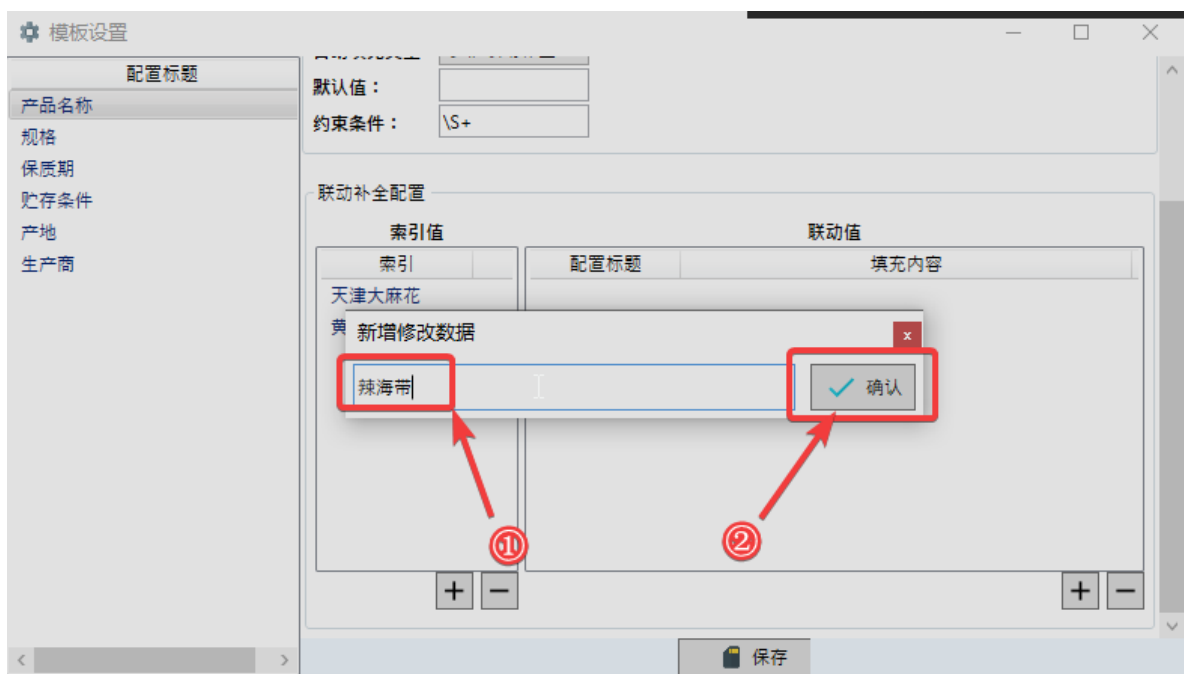
2. 在模板配置界面中左侧“配置标题”选中“产品名称”预留值



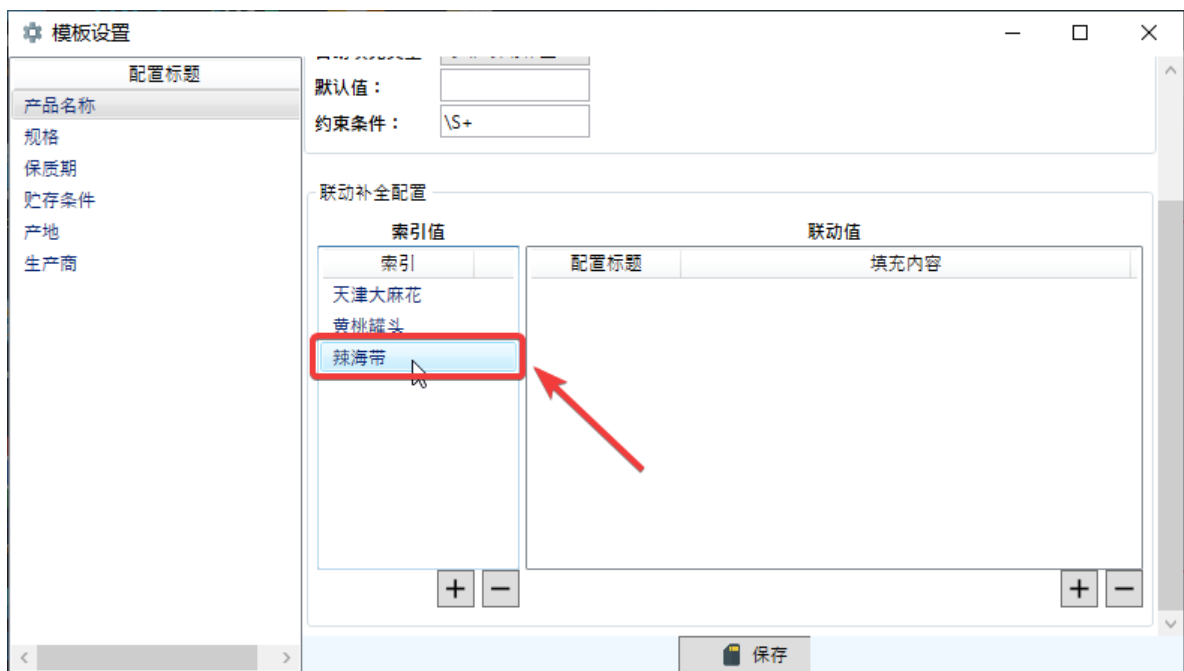
3. 点击索引值列表下方的 + 新增一个备选项



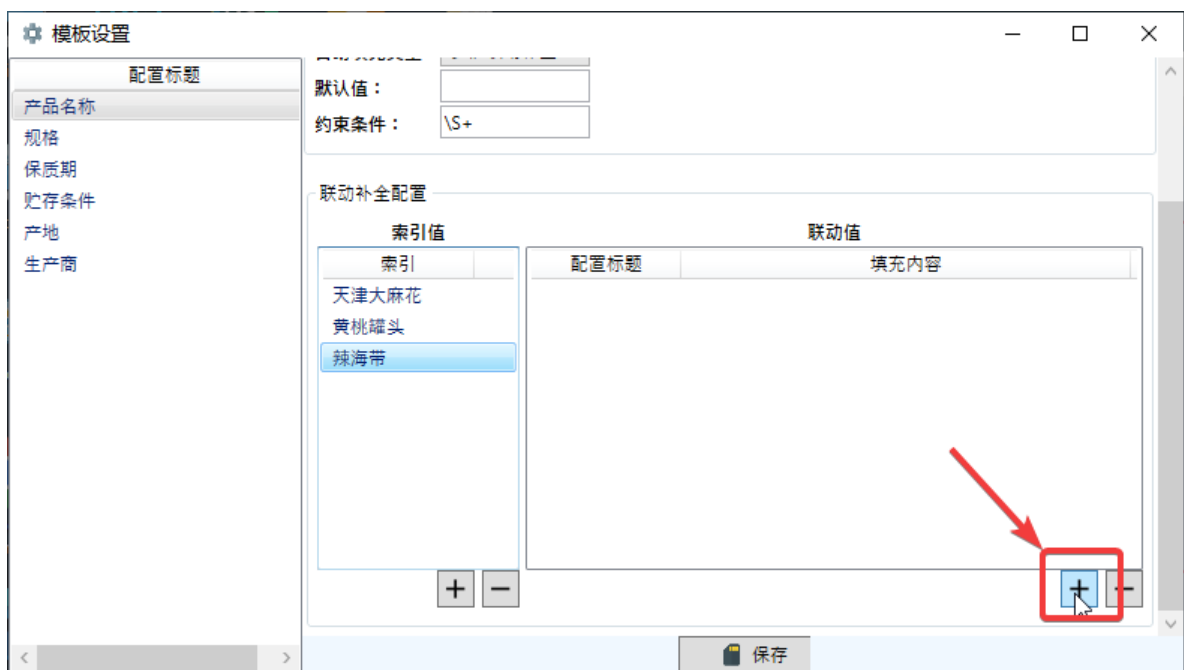
4. 在弹出的输入框中输入“辣海带”，然后点击 确认



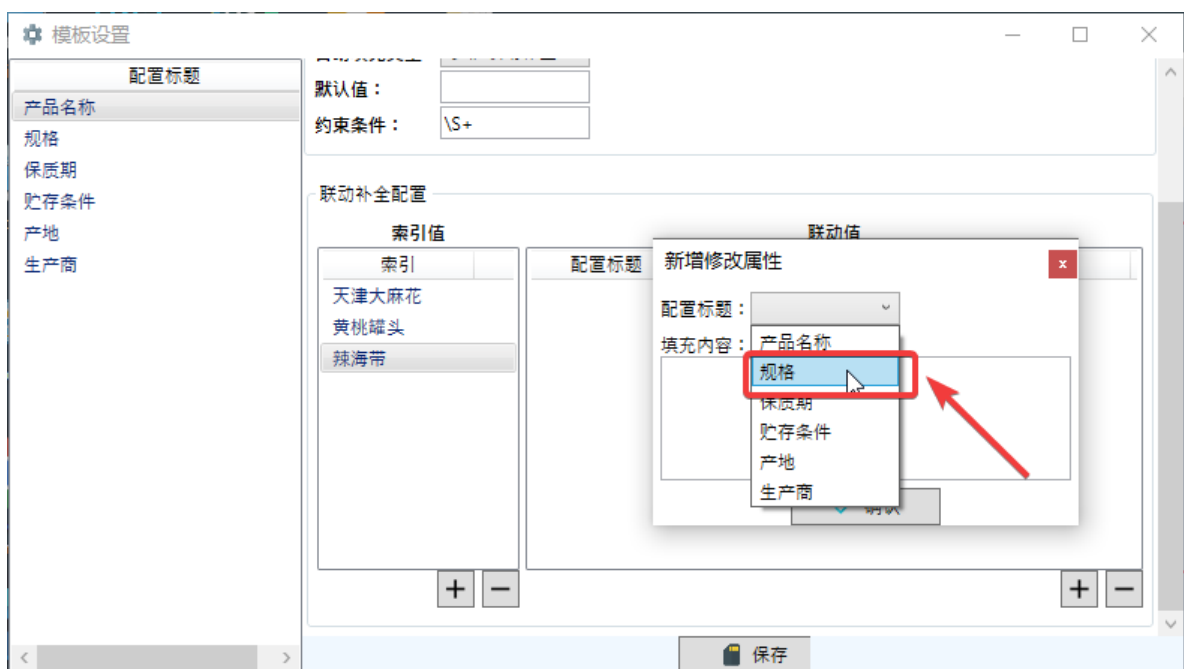
5. 在索引值列表选中刚刚添加的“辣海带”



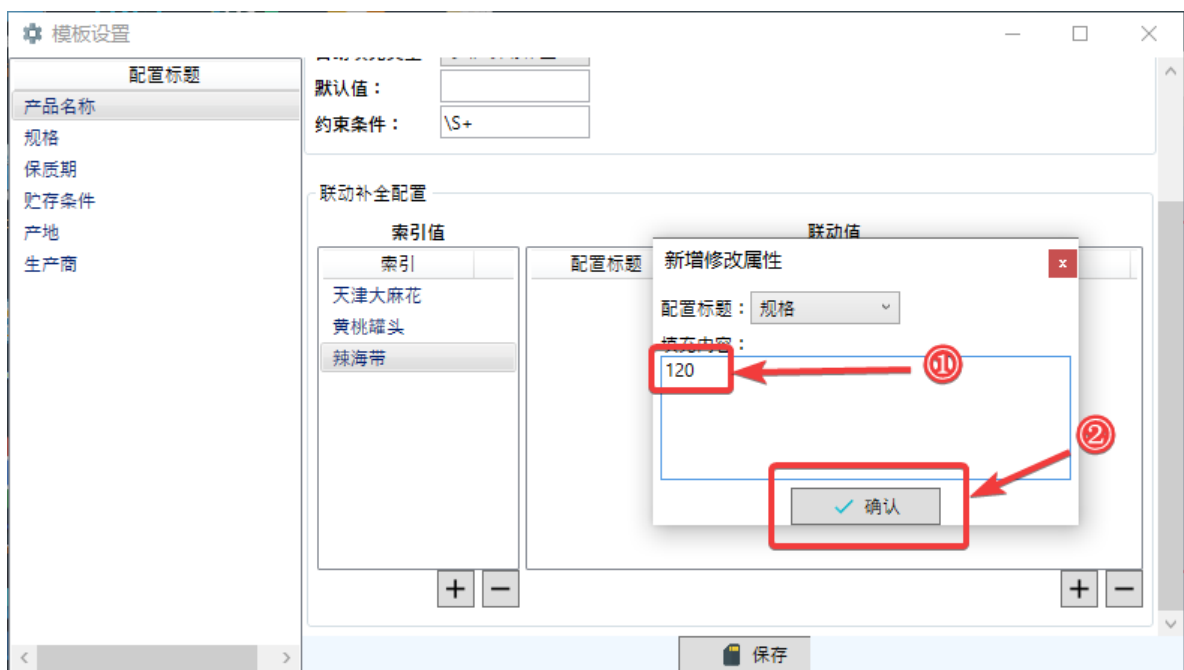
6. 在联动值右下角点击 + 新增联动值



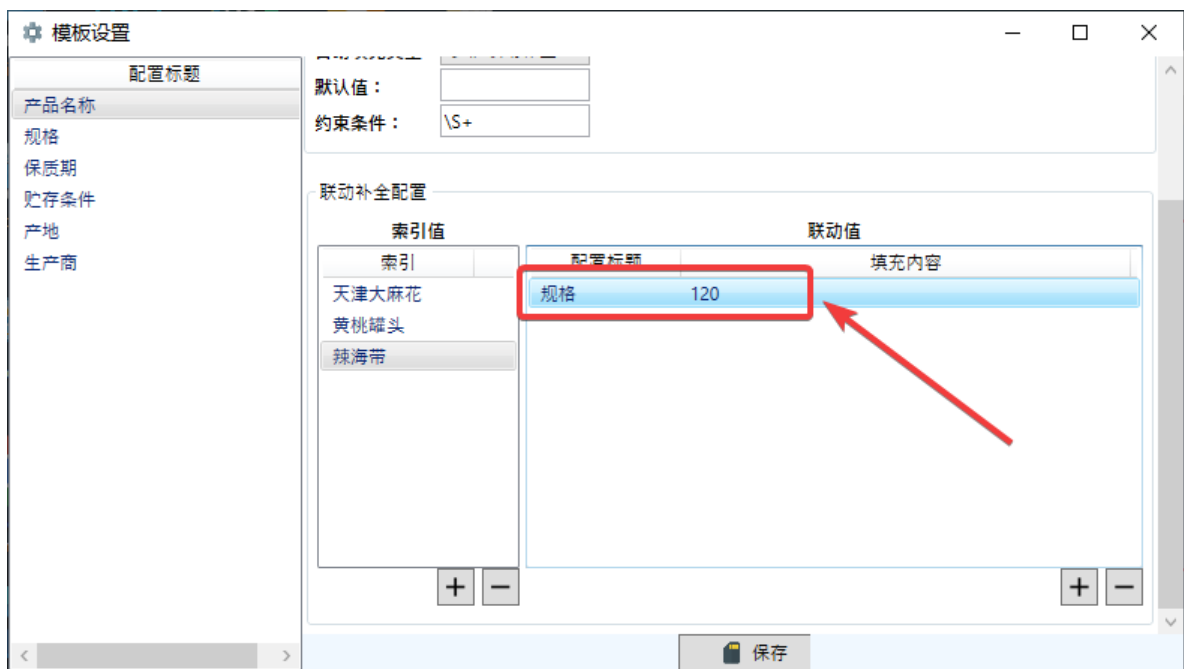
7. 在弹出窗口的 配置标题 下拉菜单中选择“规格”



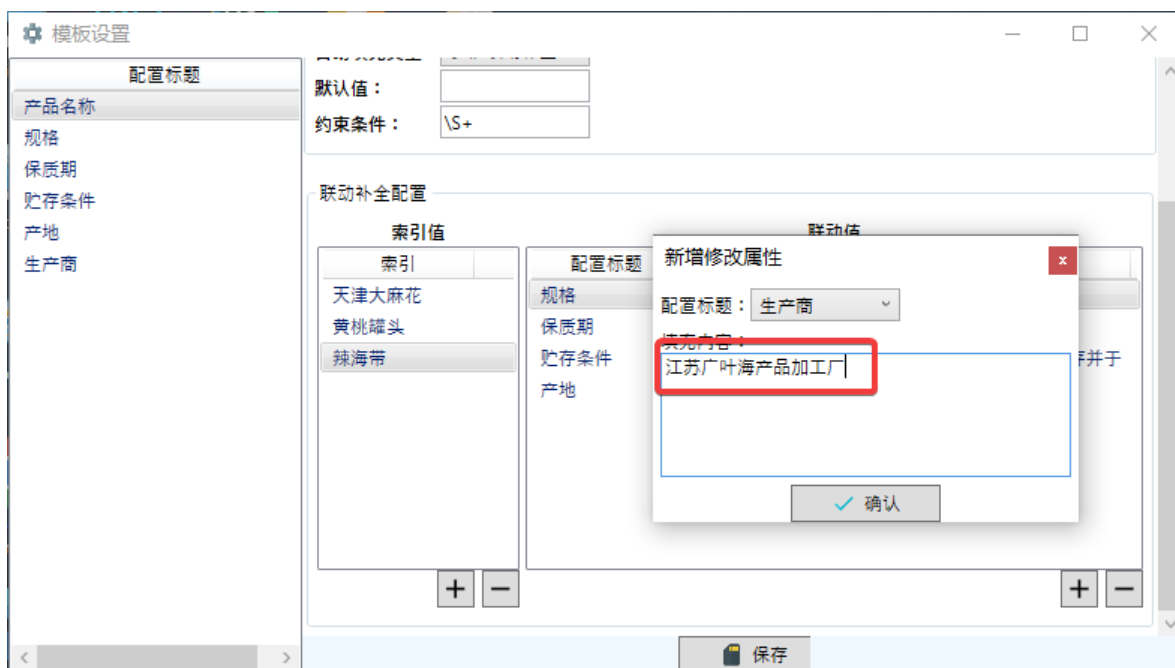
8. 输入 填充内容“120”后，点击确认保存



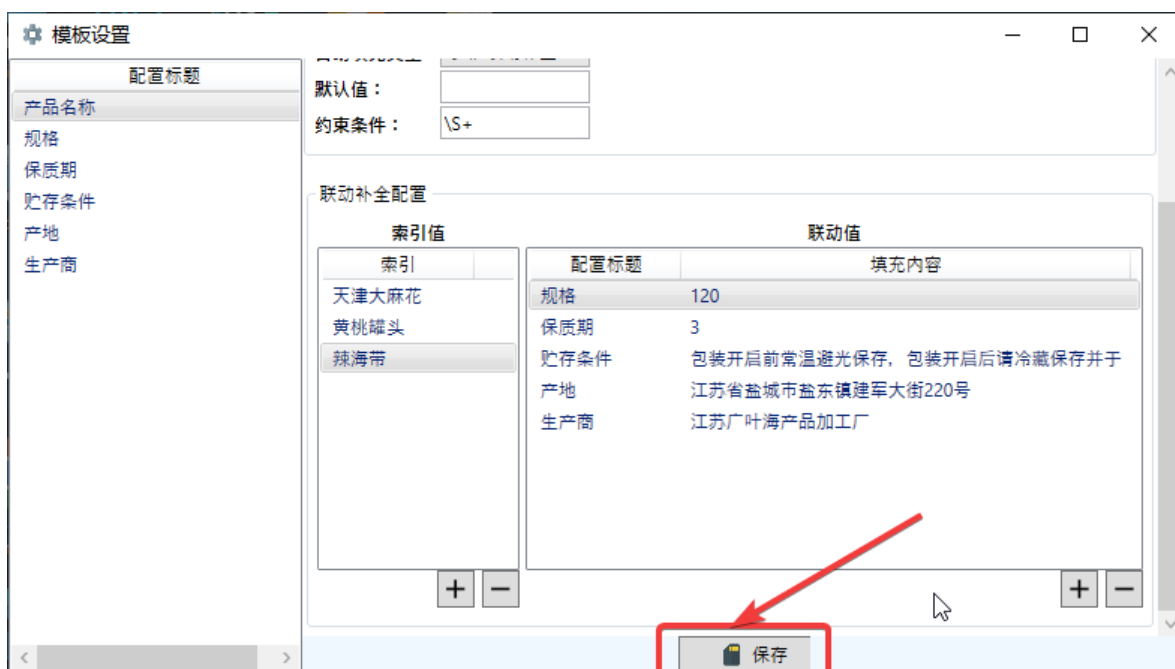
9. 至此，规格 联动值添加完毕



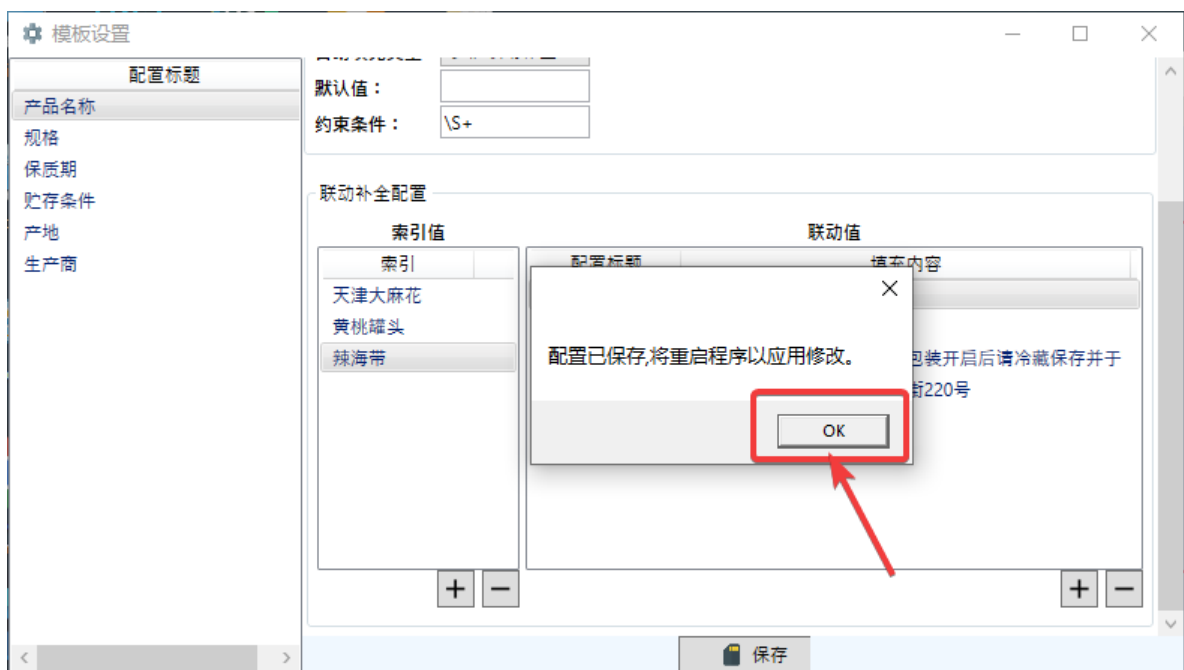
10. 使用同样的方法添加 保质期、贮存条件、产地、生产商的联动值



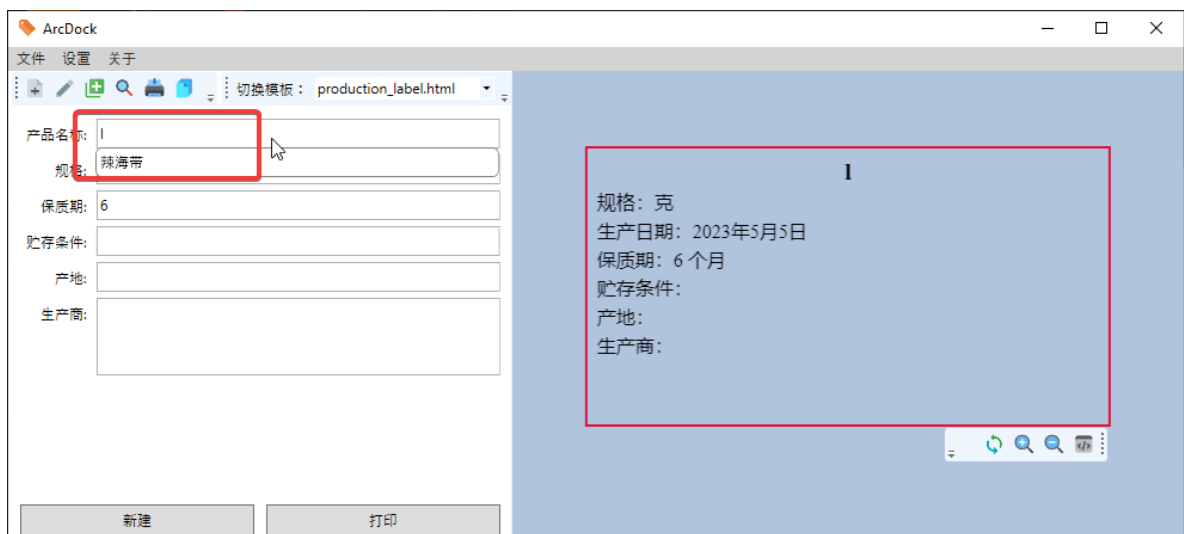
11. 确认完毕后，点击下方 保存 按钮



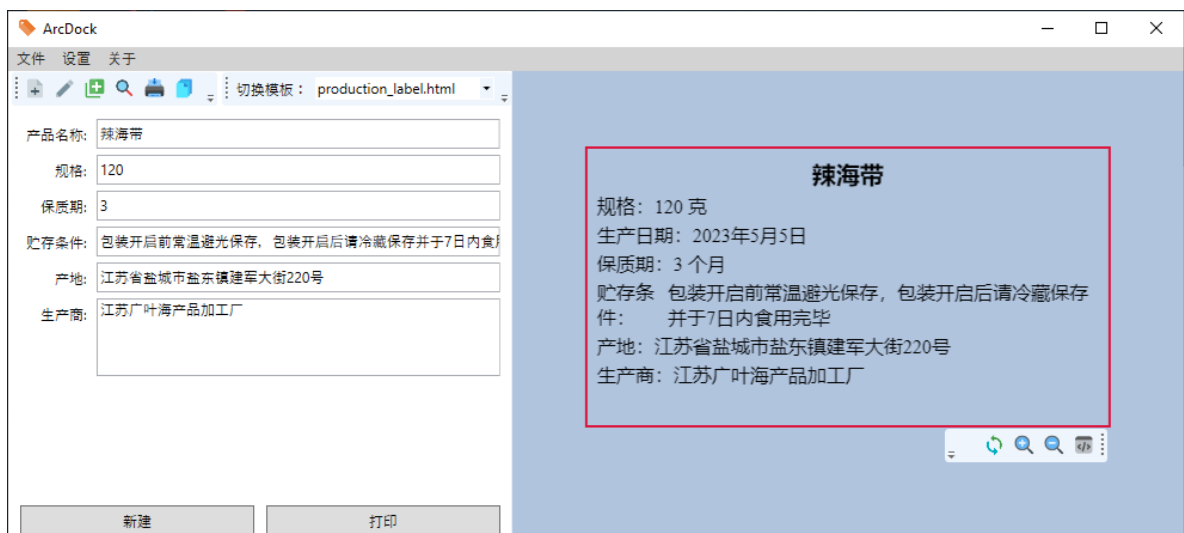
12. 提示重启应用，点击确定重启，若未唤起应用可以手动再打开



13. 程序启动后, 在 产品名称 中输入“辣海带”的拼音头“l”, 看到联想内容已成功添加。



14. 选中备选项“辣海带”, 可以看到联动内容成功填充



同步配置

权限

配置的收发方拥有的权限如下：

- 发送方：决定是否发送全局配置
- 接收方：决定是否替换本地模板、是否替换本地全局配置

同步内容：

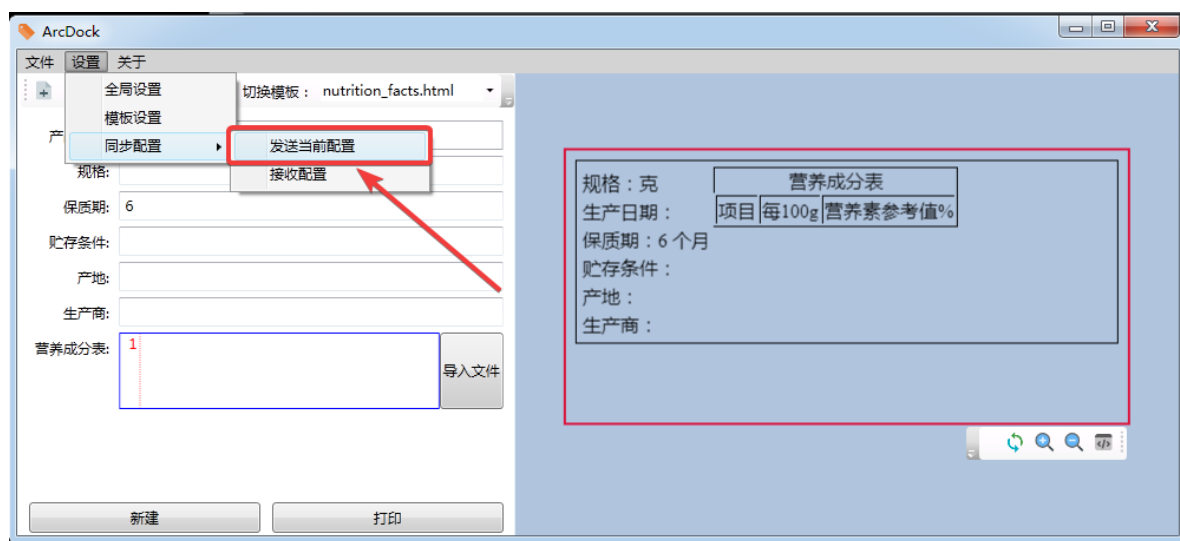
- 模板配置
 - 样式、预留值配置、备选项等
- 全局配置
 - 是否启用模板预设规则
 - 打印API设置
 - 文本分析脚本

收发流程

设要发送的数据的客户端为A，要接收数据的客户端为B。

- 客户端A启动文件发送服务

点击菜单栏 设置 → 同步配置 → 发送当前配置

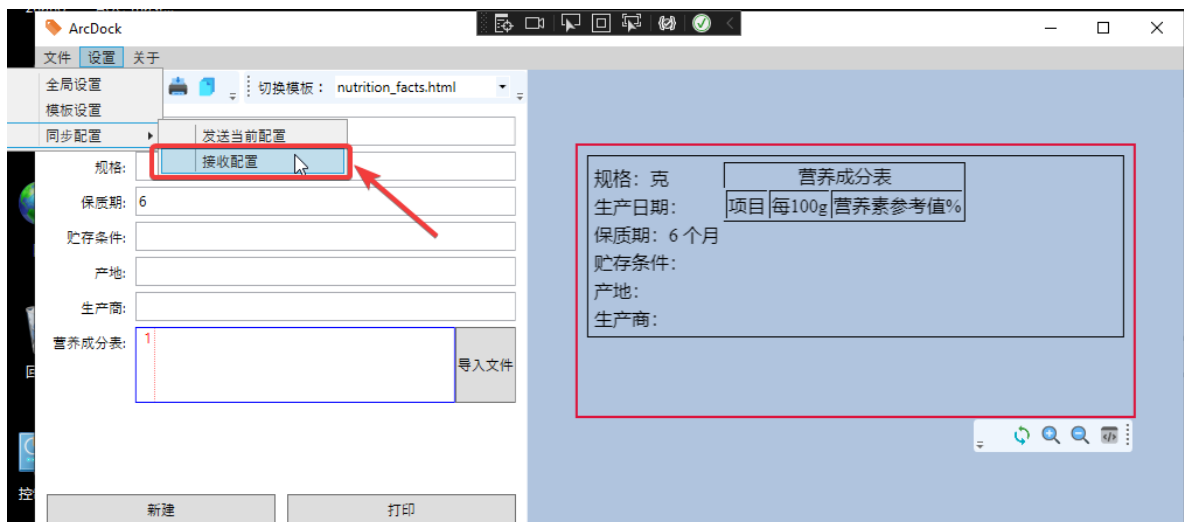


如需发送全局配置，请在发送界面勾选 发送全局配置



- 客户端B启动接收服务

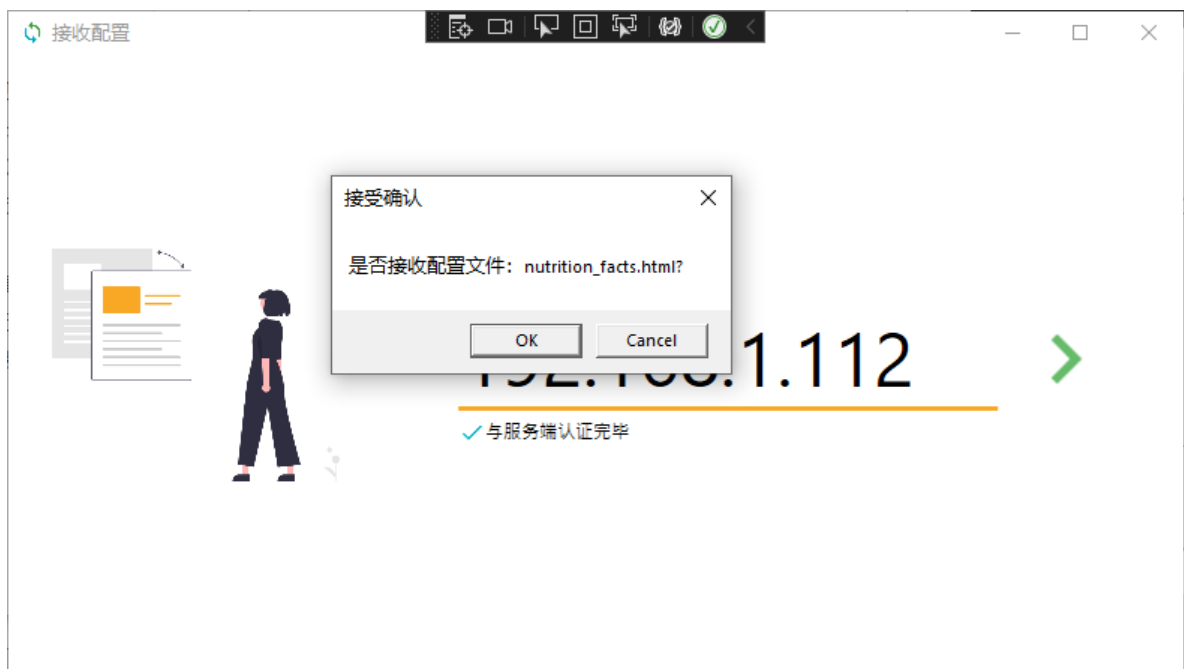
点击菜单栏 设置 → 同步配置 → 接收配置



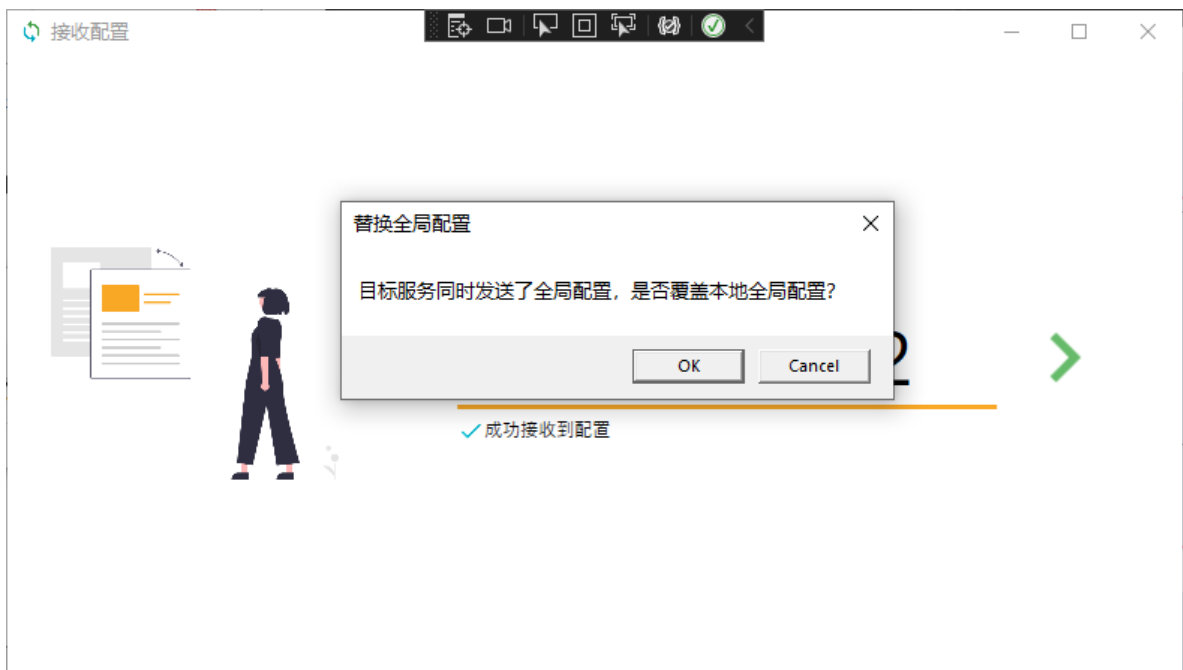
输入客户端A显示的地址，之后点击右侧 >



成功建立连接后，会提示是否接收文件，点击“确定”接收并替换本地文件



若客户端A同时勾选了“发送全局配置”，还会提示是否覆盖本地全局配置：



常见问题

- 若切换模板后，预览界面没有刷新，可以尝试填写预留值数据触发刷新。

面向维护人员

概述

作为维护人员，首先希望您能够掌握面向用户的所有章节。在此基础上，本章节增补的内容有：

- 文件结构
介绍程序目录下的各个文件、目录的作用
- 部署程序
您将了解程序的安装、部署等相关操作
- 模板结构
学习有关模板文件的相关知识，将帮助您构建模板
- 文本分析
介绍文本分析脚本的相关知识
- 交互性
您将在此学习本程序如何与其他三方程序构建交互

前置知识

维护本程序需要一些前置知识，在此默认运维人员已经掌握，不再赘述。这些技术包括：

- 基础的网页编写能力（HTML、CSS、JavaScript）

这是构建模板的基础。不过实际应用中甚少用到网页的高级功能（例如超链接、动画、AJAX等），所以您只需基础的网页编写能力即可上手。对于简单的模板，您可以无需JavaScript即可编写。但对于需要导入JSON的高级模板，需要您具备基础的JavaScript编写能力。

- 基础的编写Python脚本能力
文本分析功能使用Python解释器实现，如要使用此功能，需要您具备基础的编写Python处理字符串的能力。
- 阅读及编写JSON
JSON作为模板中的配置部分载体，有些功能无法通过软件前台界面修改（如打印机设置、资源设置等）。同时JSON也是导入文件的数据载体。您需要了解基本的JSON结构、转义方式等知识，不当的修改可能导致程序故障。
- 阅读和编写正则表达式
正则表达式作为预留值的约束条件载体，虽然对用户开放修改，但用户可能不明白如何实现。希望您具备基本的正则表达式阅读和编写能力以帮助用户定制模板。

文件结构

一览表

文件名	文件类型	重要	作用
app.publish x64 x86	目录		编译附件
template	目录	▲	模板目录
target	目录		图文媒体生成目录
Properties locales GPUCache	目录		CEF目录
Log	目录	▲	程序运行日志目录
help	目录		帮助文档目录

文件名	文件类型	重要	作用
CefSharp.BrowserSubprocess.Core.dll CefSharp.BrowserSubprocess.Core.pdb CefSharp.BrowserSubprocess.exe CefSharp.BrowserSubprocess.pdb CefSharp.Core.dll CefSharp.Core.pdb CefSharp.Core.Runtime.dll CefSharp.Core.Runtime.pdb CefSharp.Core.Runtime.xml CefSharp.Core.xml CefSharp.dll CefSharp.pdb CefSharp.Wpf.dll CefSharp.Wpf.pdb CefSharp.Wpf.xml CefSharp.xml chrome_100_percent.pak chrome_200_percent.pak chrome_elf.dll d3dcompiler_47.dll libcef.dll libEGL.dll libGLESv2.dll icudtl.dat README.txt resources.pak snapshot_blob.bin v8_context_snapshot.bin vk_swiftshader.dll vk_swiftshader_icd.json vulkan-1.dll LICENSE.txt	依赖包		CEF包
debug.log	日志		CEF调试日志 (JavaScript运行时日志)

文件名	文件类型	重要	作用
EntityFramework.dll EntityFramework.SqlServer.dll EntityFramework.SqlServer.xml EntityFramework.xml BouncyCastle.Crypto.xml BouncyCastle.Crypto.dll Microsoft.Dynamic.dll Microsoft.Dynamic.xml Microsoft.mshtml.dll Microsoft.Scripting.dll Microsoft.Scripting.Metadata.dll Microsoft.Scripting.Metadata.xml Microsoft.Scripting.xml Microsoft.Xaml.Behaviors.dll Microsoft.Xaml.Behaviors.pdb Microsoft.Xaml.Behaviors.xml System Buffers.dll System Buffers.xml System.Memory.dll System.Memory.xml System.Numerics.Vectors.dll System.Numerics.Vectors.xml System.Runtime.CompilerServices.Unsafe.dll System.Runtime.CompilerServices.Unsafe.xml	依赖包		二级依赖包
System.Data.SQLite.dll System.Data.SQLite.EF6.dll System.Data.SQLite.Linq.dll System.Data.SQLite.xml	依赖包		SQLite数据库
history.db	数据库	▲	SQLite数据库文件
IronPython.dll IronPython.Modules.dll IronPython.Modules.xml IronPython.SQLite.dll IronPython.SQLite.xml IronPython.Wpf.dll IronPython.Wpf.xml IronPython.xml	依赖包		IronPython解释器引擎
AutoCompleteTextBox.pdb AutoCompleteTextBox.dll	依赖包		智能文本框控件

文件名	文件类型	重要	作用
ArcDock.pdb ArcDock.exe.manifest config.json ArcDock.application	主程序资源		编译生成的主程序资源
log.config log4net.dll log4net.xml	依赖包		Log4net日志生成包
Newtonsoft.Json.dll Newtonsoft.Json.xml	依赖包		JSON处理包
NPinyinPro.dll	依赖包		拼音头处理包
NPOI.dll NPOI.OOXML.dll NPOI.OOXML.pdb NPOI.OOXML.xml NPOI.OpenXml4Net.dll NPOI.OpenXml4Net.pdb NPOI.OpenXml4Net.xml NPOI.OpenXmlFormats.dll NPOI.OpenXmlFormats.pdb NPOI.pdb NPOI.xml	依赖包		Excel处理包
PDF-XChange Viewer Settings.dat PDFtoPrinter.dll PDFtoPrinter_m.exe	依赖包		PDFtoPrinter API
Spire.Pdf.dll Spire.Pdf.xml	依赖包		Spire.Pdf API
ArcDock.exe.config	配置	▲	主程序配置，包括连接字符串和全局配置
ArcDock.exe	主程序		主程序

扩展名说明

- *.config
主程序或依赖包的配置文件
- *.pdb
主程序或依赖包的编译符号文件，可以删除，但删除后不易排错
- *.log

重要文件说明

- template目录

存放模板及其资源的目录，程序会扫描该目录下的html文件并在初始化时将其装载入模板。

- Log目录

主程序日志目录，日志文件名称为 `runtime.log`，体积限定在500KB。

- history.db

SQLite数据库文件，存储打印历史和打印和Python脚本。

- ArcDock.exe.config

主程序配置文件，存储SQLite连接字符串、全局配置。

连接字符串：

```
<connectionStrings>
  <add name="history" connectionString="data source=history.db;version=3;" />
</connectionStrings>
```

默认全局配置：

```
<userSettings>
  <ArcDock.Properties.Settings>
    <setting name="UserPrintApi" serializeAs="String">
      <value>3</value>
    </setting>
    <setting name="IsEnableRules" serializeAs="String">
      <value>True</value>
    </setting>
  </ArcDock.Properties.Settings>
  <ArcDock.Settings>
    <setting name="UserPrintApi" serializeAs="String">
      <value>0</value>
    </setting>
  </ArcDock.Settings>
</userSettings>
```

部署

运行环境

- 软件环境
 - 操作系统：32位或64位Windows7 SP1及以上版本的Windows
 - 运行环境：.NET Framework 4.7.2
- 硬件环境
 - CPU：1GHz及以上

- RAM: 1G及以上
- 预留磁盘空间: 500M以上

部署方法

将程序目录导入目标电脑, 手动创建主程序 `ArcDock.exe` 的快捷方式即可。若系统未安装.NET Framework 4.7.2运行时, 请先[安装.NET Framework 4.7.2运行时](#)。

注意: 如要更新程序, 请先备份 `history.db`, 更新后替换 `history.db`。

模板结构

概述

模板文件使用**HTML文件标准**作为载体, 存储在软件目录的**template**文件夹中, **注意后缀名必须是html, 否则不会识别。**

模板内部分为两部分:

- 配置部分

配置部分使用 `<script type="config/json"></script>` 标签引入, 配置部分定义了模板的各种配置, 如默认打印机、模板外观的大小、预留值和自动填充值等参数, 是一段JSON字符串。一个模板文件只能有一个配置标签, 配置标签可以出现在符合HTML标准的任意位置, 不过习惯上常放在 `<head></head>` 标签内。

- 样式部分

除了配置部分其他都是样式部分, 样式部分定义了配置文件的排版等外观配置。

虽然模板使用HTML文件标准, 可以作为普通网页解析, 但由于读取模板时先按照XML标准读取配置再从CEF中渲染, 某些仅HTML可用、XML不可用的标准不被支持, 例如:

- **不支持单标签**

不支持诸如 `meta / br / input` 等单标签, 如要使用单标签功能请用其双标签格式。

- **JavaScript代码块需转义**

JavaScript出现XML需转义内容, 如"`</>`"等, 会被转义为XML, 由此建议在JavaScript代码块中使用 `<![CDATA[CODE]]>` 注释标签, 或者使用外链JavaScript文件。如:

```
<div id="date"></div>
<script>
  <![CDATA[
    let date = new Date()
    document.getElementById("date").innerText = date.getFullYear()+"年"+
    (date.getMonth()+1)+"月"+date.getDate()+"日"
  >
</script>
```

配置部分

配置部分的JSON由两部分组成，分别是**文档配置 (settings)** 和**数据配置 (data)**，结构如下：

```
{
  "setting": {
    ...
  },
  "data": [
    {}, {}, {}, ....
  ]
}
```

文档配置主要设置一些与页面信息有关的配置，如页面大小、打印大小、默认打印机等，格式如下：

```
"settings": {
  "printer": "", //默认打印机，空串则由系统指定
  "height": 265, //页面高度，单位是像素
  "width": 340, //页面宽度，单位是像素
  "print_unit": "cm", //打印宽高的单位，目前没有实现这个配置，默认都是厘米
  "print_height": 7, //打印高度，有的API不受该配置控制
  "print_width": 9, //打印宽度，有的API不受该配置控制
  "zoom": 0, //模板内放大参数，目前没有实现这个配置
  "resource": ["bg.jpg"] //模板资源，如果模板有引用到其他文件，请在此注册资源。本数组存放资源
  路径，可以是相对模板文件路径或绝对路径
}
```

数据配置是一个列表，列表内的每一项都是预留值的信息。预留值的信息模板如下：

```
{
  "id": "medicament_num", //预留值的名称，也就是渲染页面中{{value}}内部包裹的value名称
  "name": "药品数量", //预留值友好名称，显示在程序中让用户填写的文本框标题
  "type": "input", //预留值类型，目前支持三种类型：普通文本框input、智能文本框autoinput、多
  行文本框richinput、JSON对象json
  "opt_type": 0, //文本框的补全选项类型，0为无自动补全，普通文本框和多行文本框该值都应该为0，1
  为单框文本补全，2为多框联动补全
  "option": [], //当上面的opt_type为1时，检索其中的内容并给用户提示。参见单款文本补全的示例，
  当type为json时，该列表是文件解析对应的列ID
  "option_exc": [], //当上面的opt_type为1时，检索其中的内容并给用户提示，并根据用户选中的内容
  自动填充其他预留值，参见多框联动补全的示例
  "default": "", //进入系统时该预留值默认填充的值
  "rules": "[0-9]" //检查规则，是区分大小写的正则表达式，只有正则测试通过才允许用户打印。
}
```

案例

单框文本补全

用户输入值时给用户的备选项，输入拼音头或者文本都可以触发提示，用户选中后填充预留值。其 opt_type 为1，备选内容以字符串列表的形式填写在 option 中。例如下面的预留值，用户输入“神”、“神经内”、“S”、“SJN”等都可以提示神经内科。

```
{
  "id": "patient_dept",
  "name": "病人科室",
  "type": "autoinput",
  "opt_type": 1,
  "option": [
    "肿瘤科",
    "神经内科",
    "神经外科",
    "小儿外科",
    "创伤中心"
  ],
  "option_exc": [],
  "default": "",
  "rules": "\\S+"
}
```

多框联动补全

用户输入值时给用户的备选项，输入拼音头或者文本都可以触发提示，用户选中后填充预留值，**并且触发填写其他预留值**。其 opt_type 为2，备选内容以对象列表的形式填写在 option_exc 中。option_exc 的对象示例如下：

```
{
  "content": "乳清", //目前预留值的填充内容，与option中的字符串效果相同
  "exec": [ //选中该项备选项时，还应该给其他预留值填充的内容
    {
      "key": "medicament_unit", //填充另一个预留值的ID
      "content": "桶" //填充另一个预留值的内容
    }, {
      "key": "medicament_usage",
      "content": "测试内容"
    }
  ]
}
```

这段JSON表示当用户选择“乳清”这个备选项时，不仅为当前预留值填充“乳清”，也为ID为 medicament_unit 的预留值填充内容“桶”，为ID为 medicament_usage 的预留值填充“测试内容”

JSON对象

导入JSON对象需要在文档配置和样式配置配合实现。

首先需要在预留值配置的 option 列表添加需要解析的对象名，用户导入文件后软件将数据转换为JSON。如对于一个预留值配置：

```
{
  "id": "obj1",
  "name": "对象1",
  "type": "json",
  "opt_type": 0,
  "option": ["id", "content"],
  "option_exc": [],
  "default": "",
  "rules": ""
},
```

用户导入一个Excel文件：

名称	产量
苹果	50斤
橘子	100斤

并且在软件中指定ID对应名称，content对应产量，

由此将得到JSON：

```
{
  "id": ["苹果", "橘子"],
  "content": ["50斤", "100斤"]
}
```

程序将用此JSON去替换预留值obj1的值，所以我们需要编写JavaScript解析它：

```
<div id="display"></div>
<!--在文档最后添加-->
<script>
  //<![CDATA[
    let val = {{obj1}} //等价于let val = {"id":["苹果","橘子"],"content":["50斤","100斤"]}
    document.getElementById("display").innerText = val.id[0] //在display中显示了"苹果"
  //]]>
</script>
```

文本分析

原理

本程序内容嵌入了一个Python解释器用于文本分析，您可编写Python脚本分析文本。

输入值：字符串类型 `source`

输出值：字符串字典 `result`

示例：

当输入字符串为“abc,1234”时，以“,”进行分割，将切分后的“abc”赋给ID为 val1 的预留值，“1234”赋给ID为 val2 的预留值。

```
result['val1'] = source.split(',')[0] #将输入值按照","进行拆分，将拆分后第一个值赋给val1
result['val2'] = source.split(',')[1] #将输入值按照","进行拆分，将拆分后第二个值赋给val2
```

编写脚本

目标

完成本文中[使用方法](#)>[工作流程](#)>[文本解析](#)的分析方法。

具体为分析下列文本：

天津大麻花|500|6| 常温避光保存|天津市宁河区芦台街道光明路107号|天津市国营食品生产厂

按照“|”进行分割，使得分析结果为：

产品名称 = 天津大麻花

规格 = 500

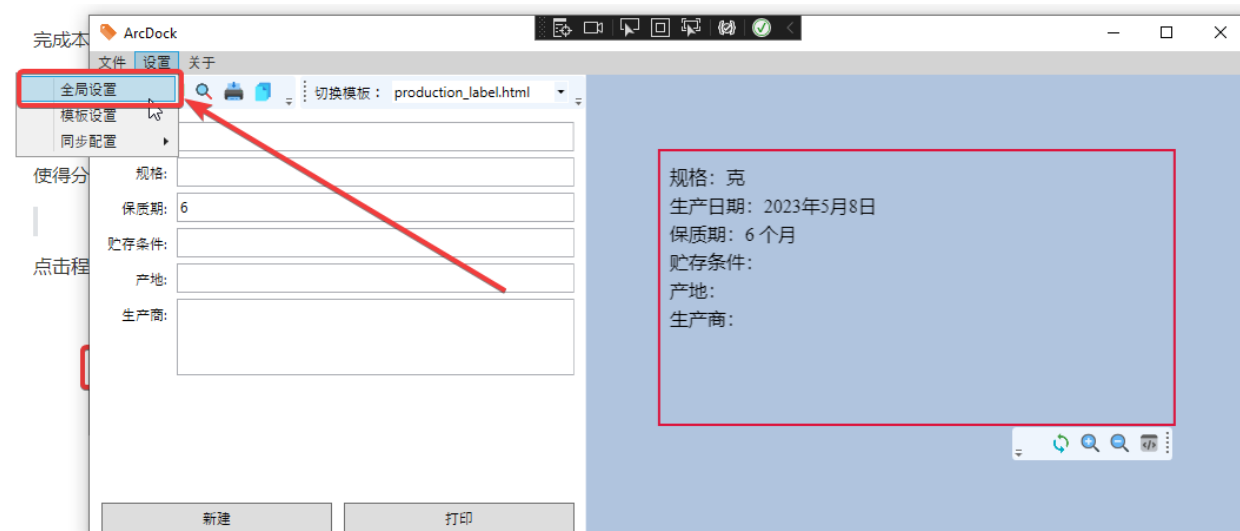
保质期 = 6

贮存条件 = 常温避光保存

产地 = 天津市宁河区芦台街道光明路107号

生产商 = 天津市国营食品生产厂

切换到相应模板，点击程序菜单栏的 设置 → 全局设置



在弹出窗口中切换到 python 选项卡，点击 文本解析 区域内的 代码编辑器



代码编辑器打开后界面如下：



首先将输入值source按照“|”进行分割，如下：

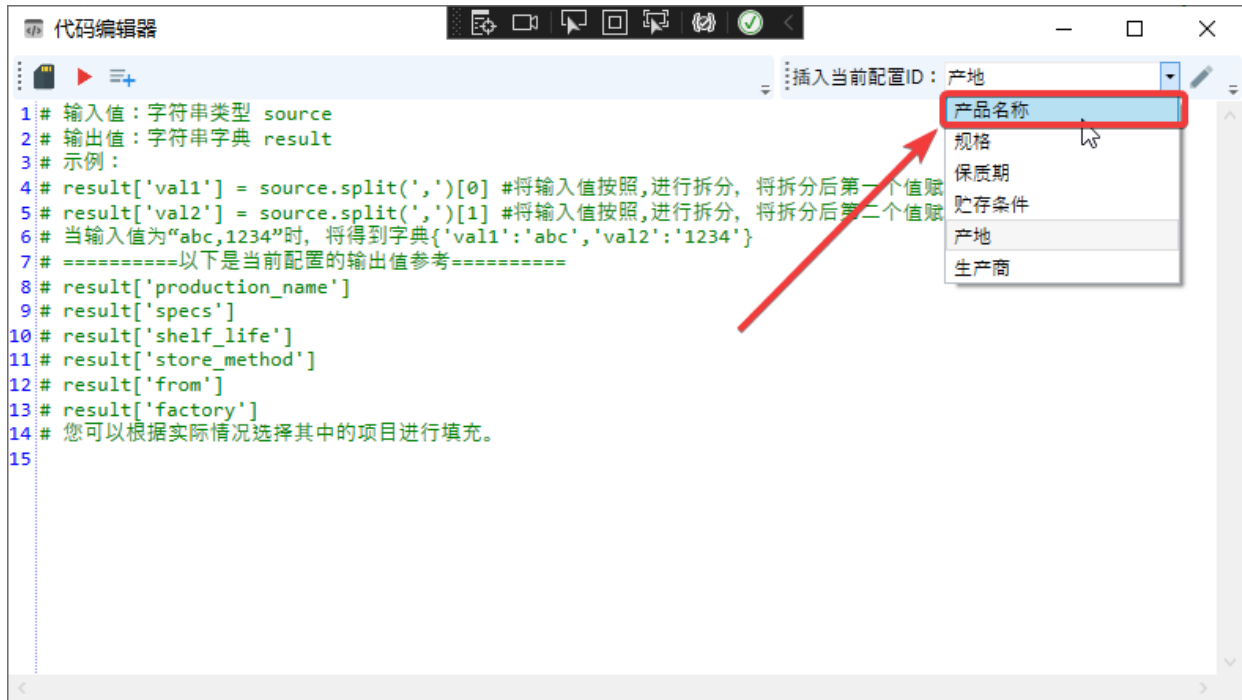
```
array = source.split('|') #按照“|”进行分割
```

首先给产品名称赋值，您可以手动编写以下代码：

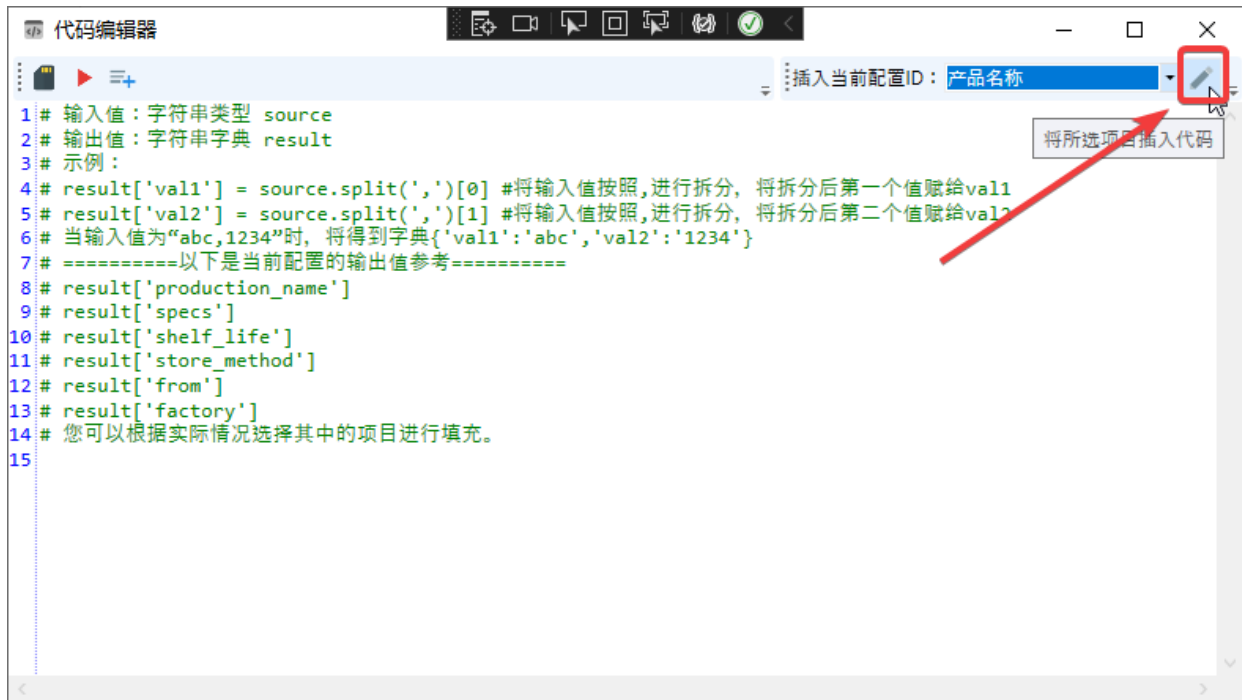

```
result['production_name'] = array[0] #给产品名赋值
```

也可以使用下面的方法快速赋值：

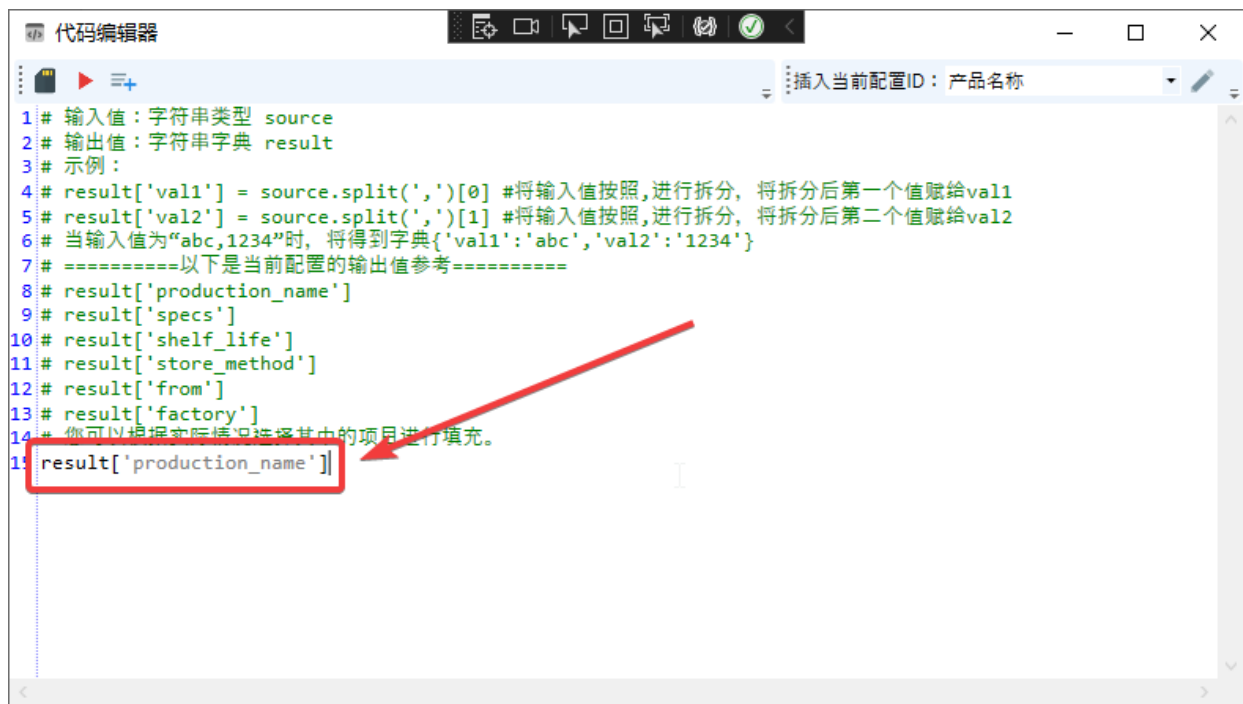
在代码编辑器界面右上角找到 插入当前配置ID 下拉列表框，选择产品名称：



点击旁边的 将所选项目插入代码 按钮：



即可快速插入预留值ID字典



```
1 # 输入值：字符串类型 source
2 # 输出值：字符串字典 result
3 # 示例：
4 # result['val1'] = source.split(',')[0] #将输入值按照,进行拆分，将拆分后第一个值赋给val1
5 # result['val2'] = source.split(',')[1] #将输入值按照,进行拆分，将拆分后第二个值赋给val2
6 # 当输入值为“abc,1234”时，将得到字典{'val1':'abc','val2':'1234'}
7 # =====以下是当前配置的输出值参考=====
8 # result['production_name']
9 # result['specs']
10 # result['shelf_life']
11 # result['store_method']
12 # result['from']
13 # result['factory']
14 # 您可以根据实际情况选择其中的项目进行填充。
15 result['production_name']
```

然后将代码补全即可



```
1 # 输入值：字符串类型 source
2 # 输出值：字符串字典 result
3 # 示例：
4 # result['val1'] = source.split(',')[0] #将输入值按照,进行拆分，将拆分后第一个值赋给val1
5 # result['val2'] = source.split(',')[1] #将输入值按照,进行拆分，将拆分后第二个值赋给val2
6 # 当输入值为“abc,1234”时，将得到字典{'val1':'abc','val2':'1234'}
7 # =====以下是当前配置的输出值参考=====
8 # result['production_name']
9 # result['specs']
10 # result['shelf_life']
11 # result['store_method']
12 # result['from']
13 # result['factory']
14 # 您可以根据实际情况选择其中的项目进行填充。
15
16 array = source.split(',') #按照“,”进行分割
17 result['production_name'] = array[0]
```

使用相同的方式给所有预留值赋值：

```
1 # 输入值：字符串类型 source
2 # 输出值：字符串字典 result
3 # 示例：
4 # result['val1'] = source.split(',')[0] #将输入值按照,进行拆分,将拆分后第一个值赋给val1
5 # result['val2'] = source.split(',')[1] #将输入值按照,进行拆分,将拆分后第二个值赋给val2
6 # 当输入值为“abc,1234”时,将得到字典{'val1':'abc','val2':'1234'}
7 # =====以下是当前配置的输出值参考=====
8 # result['production_name']
9 # result['specs']
10 # result['shelf_life']
11 # result['store_method']
12 # result['from']
13 # result['factory']
14 # 您可以根据实际情况选择其中的项目进行填充。
15
16 array = source.split(',') #按照“,”进行分割
17 result['production_name'] = array[0]
18 result['specs'] = array[1]
19 result['shelf_life'] = array[2]
20 result['store_method'] = array[3]
21 result['from'] = array[4]
22 result['factory'] = array[5]
```

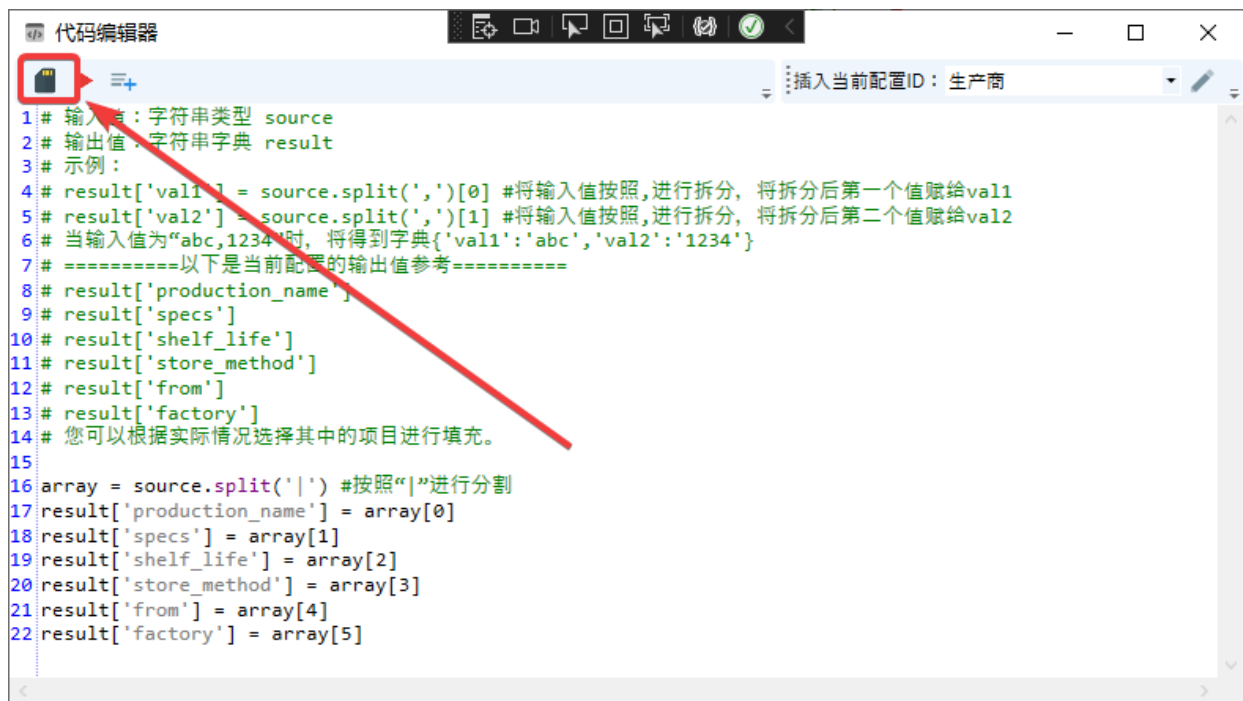
编写完毕，下面我们可以测试一下代码运行效果，点击代码编辑器窗口工具栏的 运行测试 按钮

```
1 # 输入值：字符串类型 source
2 # 输出值：字符串字典 result
3 # 示例：
4 # result['val1'] = source.split(',')[0] #将输入值按照,进行拆分,将拆分后第一个值赋给val1
5 # result['val2'] = source.split(',')[1] #将输入值按照,进行拆分,将拆分后第二个值赋给val2
6 # 当输入值为“abc,1234”时,将得到字典{'val1':'abc','val2':'1234'}
7 # =====以下是当前配置的输出值参考=====
8 # result['production_name']
9 # result['specs']
10 # result['shelf_life']
11 # result['store_method']
12 # result['from']
13 # result['factory']
14 # 您可以根据实际情况选择其中的项目进行填充。
15
16 array = source.split(',') #按照“,”进行分割
17 result['production_name'] = array[0]
18 result['specs'] = array[1]
19 result['shelf_life'] = array[2]
20 result['store_method'] = array[3]
21 result['from'] = array[4]
22 result['factory'] = array[5]
```

在弹出的代码测试器窗口中输入测试内容，点击 运行测试 按钮，即可预览预留值ID与测试文本的对应测试结果。



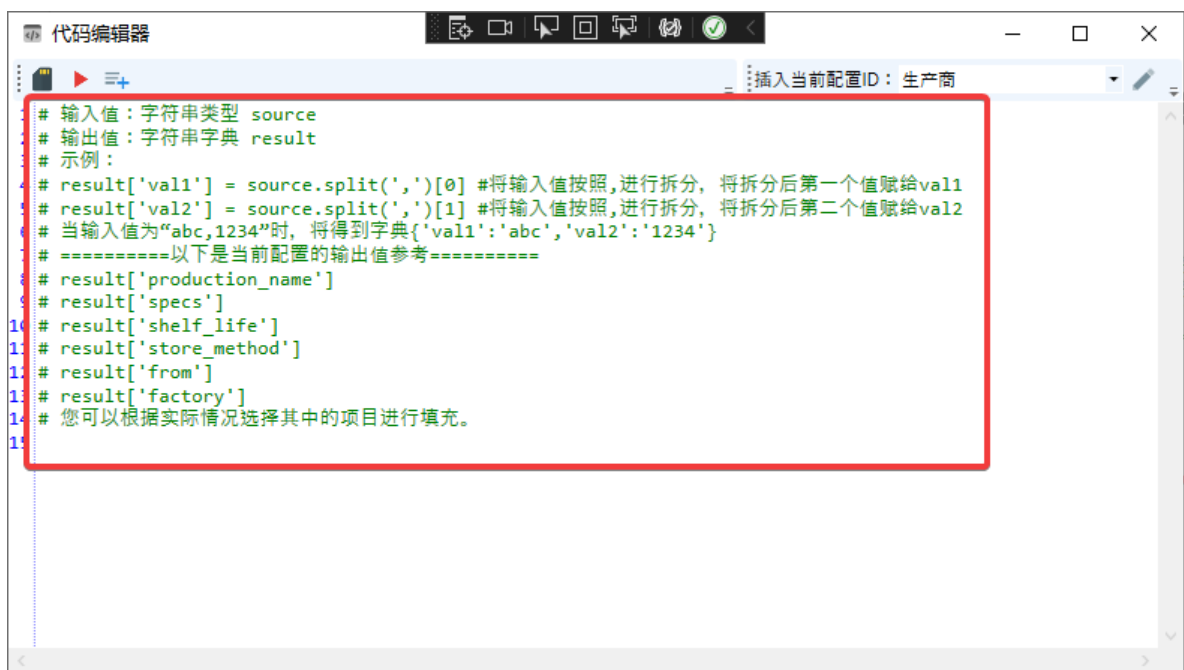
如果测试结果符合预期，即可在代码编辑器窗口中保存代码。



保存后，即可使用文本分析填充预留值。

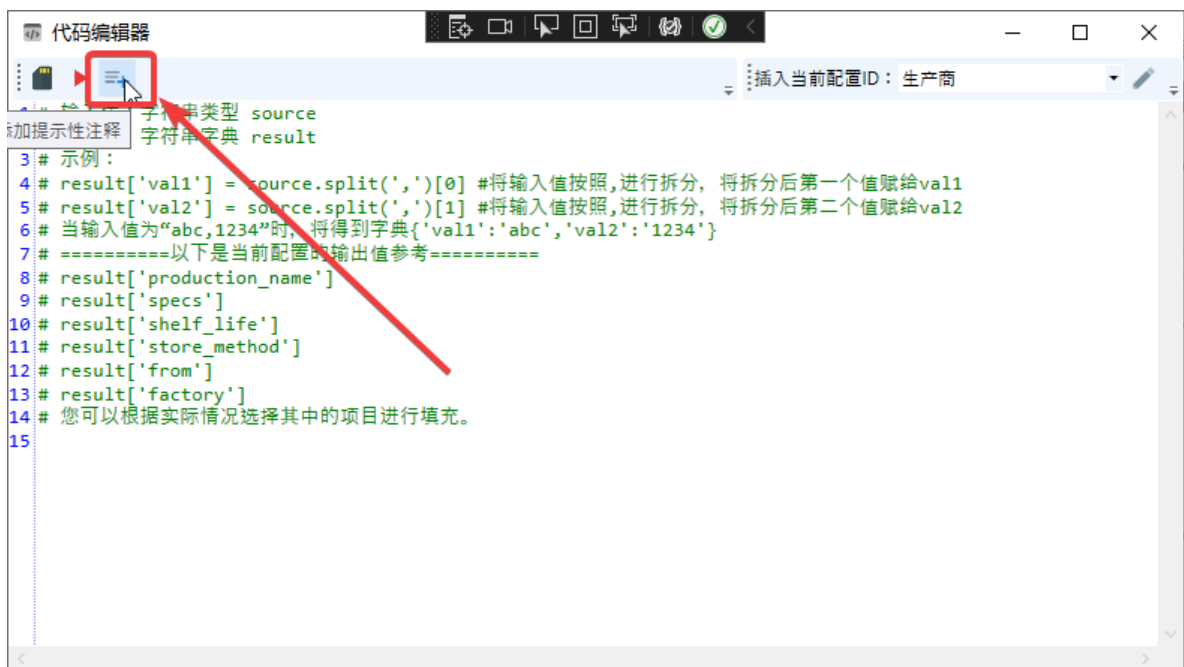
注意事项

- 插入当前配置ID使用的是当前模板文件的预留值，所以在编写脚本前请切换到正确的模板。
- 初次进入代码编辑器会根据当前模板文件自动添加提示性注释，注释可以删除，不影响代码运行结果。



```
代码编辑器
# 输入值：字符串类型 source
# 输出值：字符串字典 result
# 示例：
# result['val1'] = source.split(',')[0] #将输入值按照,进行拆分,将拆分后第一个值赋给val1
# result['val2'] = source.split(',')[1] #将输入值按照,进行拆分,将拆分后第二个值赋给val2
# 当输入值为"abc,1234"时,将得到字典{'val1':'abc','val2':'1234'}
# =====以下是当前配置的输出值参考=====
# result['production_name']
# result['specs']
# result['shelf_life']
# result['store_method']
# result['from']
# result['factory']
# 您可以根据实际情况选择其中的项目进行填充。
```

如果需要手动添加提示性注释，可以点击工具栏的 添加提示性注释 按钮



```
代码编辑器
# 输入值：字符串类型 source
# 输出值：字符串字典 result
# 示例：
# result['val1'] = source.split(',')[0] #将输入值按照,进行拆分,将拆分后第一个值赋给val1
# result['val2'] = source.split(',')[1] #将输入值按照,进行拆分,将拆分后第二个值赋给val2
# 当输入值为"abc,1234"时,将得到字典{'val1':'abc','val2':'1234'}
# =====以下是当前配置的输出值参考=====
# result['production_name']
# result['specs']
# result['shelf_life']
# result['store_method']
# result['from']
# result['factory']
# 您可以根据实际情况选择其中的项目进行填充。
```

- 无需保存脚本即可使用 运行测试，最好在运行测试通过后再保存脚本。
- 只有预留给ID与result字典都存在的ID才会被对应填充，对应关系行为标记如下：

	预留给ID存在	预留给ID不存在
result字典存在Key	对应填充	忽略
result字典不存在Key	忽略	--

交互性

CLI传参

不论程序是否已经实现启动都可以使用CLI传参，若程序未启动将启动程序并执行相应命令，若程序已启动则将程序唤醒到前台并执行相应命令（也可以使用静默打印，则可不将程序唤醒到前台）。

命令参数：

```
ArcDock [--silent][--template file_name] --args json_string
```

参数说明：

- silent
可选参数，静默打印，不需要用户点击打印直接打印图文媒体。若不启用此开关仅唤起程序传递参数，由用户选择是否打印。
- template
可选参数，选择模板文件名称，需要带文件扩展名。模板必须在程序启动时成功加载。若不启动此开关则向当前使用的模板传递数据。
- args
必选参数，数据对象，是JSON字符串，以{"预留值ID":"数据内容"}的格式传递数据。

注意：json_string是JSON字符串，由于JSON字符串中含有双引号"，在控制台中需要对其进行转义，如对于原始内容为{"key":"hello"}的JSON字符串，转义后应写为：{"\"key\":\"hello\""}。

示例：

现有预留值ID为key、content的模板名称为test.html，使用命令为其传递数据使得key=Hello，content=World，并且传参后直接打印：

```
arcdock --silent --template test.html --args  
{\"key\":\"Hello\",\"content\":\"world\"}
```

向目前使用的模板传递数据，使得key=Hello，让用户选择是否打印：

```
arcdock --args {\"key\":\"Hello\"}
```

仅将程序唤醒到前台：

```
arcdock --args {}
```