# HiBot 企业流程自动化设计平台 V1.0
# 操作手册

目录

# 1. 基本概念

本章主要介绍 Hibot 的四个基本概念：流程、流程块、命令、属性。这四个概念，贯穿本文的始终，在后面的章节中，也会反复的使用这四个概念作为基本术语。

## 1.1 概念

这四个基本概念之间都是包含关系，一个流程包含多个流程块，一个流程块包含多个命令，一个命令包含多个属性。

流程
流程块
命令
属性

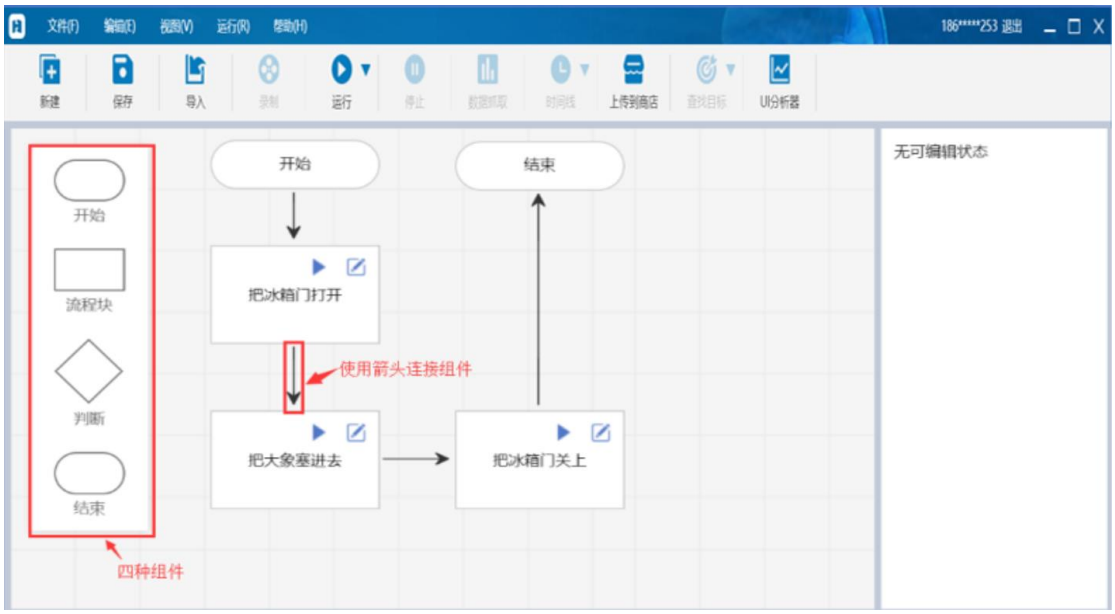
第一个概念是流程，是指要用 Hibot 来完成的一项任务，一个任务对应一个流程。虽然可以用 Hibot 陆续建立多个流程，但同一时刻，只能编写和运行一个流程。Hibot 中的流程，都是采用流程图的方式来显示的。

Hibot 内置了一些经典流程的范例，初学者可以打开这些流程范例，试试运行这些流程，进行仿照学习，当然也可以自己新建一个流程。

## 1.2 流程图

新建或打开 Hibot 中的流程后，可以看到，每个流程都用一张流程图来表示。

在流程图中，包含了“开始”、“流程块”、“判断”和“结束”四种组件，它们之间是用箭头连起来的。如图：




Hibot 的流程图

每个流程图中必须有一个、且只能有一个“开始”组件。顾名思义，流程从这里开始运行， 并且沿着箭头的指向，依次运行到后续的各个组件。

每个流程图中，可以有一个或多个“结束”组件，流程一旦运行遇到“结束”组件，自然就会停止运行。当然也可以没有“结束”组件，当流程运行到某个流程块，而这个流程块没有箭头指向其它流程块时，流程也会停止运行。

每个流程图中，可以有一个或多个“判断”组件，当然也可以没有“判断”组件。在流程运行的过程中，“判断”组件将根据一定的条件，使后面的运行路径产生分叉。条件为真的时候，沿着“yes”箭头运行后续组件；否则，沿着“no”箭头运行后续组件。

我们可以把一个任务分为多个步骤来完成，其中的每个步骤，在 Hibot 用一个“流程块”来描述。比如，假设我们的任务是“把大象装进冰箱里”，那么，可以把这个任务分为三个步骤：

- 把冰箱门打开
- 把大象塞进去
- 把冰箱门关上

上述每个步骤就是一个流程块。当然，这个例子只是打个比方，Hibot 并不能帮我们把冰箱门打开。但通过这个例子可以看出，在 Hibot 中，一个步骤，或者说一个流程块，只是大体上描述了要做的事情，而不涉及到如何去做的细节。

Hibot 并没有规定一个流程块到底要详细到什么程度：流程块可以很粗，甚至一个流程里面甚至可以只有一个流程块，在这种情况下，流程和流程块实际上已经可以看作是同一个概念了；流程块也可以很细，把一个流程拆分成很多流程块。那么究竟拆分成多少个最合适？这取决于您的个人喜好。但是，我们有两个

建议：一是把相对比较独立的流程逻辑放在一个流程块里；二是流程块的总数不宜太多，一个流程中最好不要超过 20 个流程块。

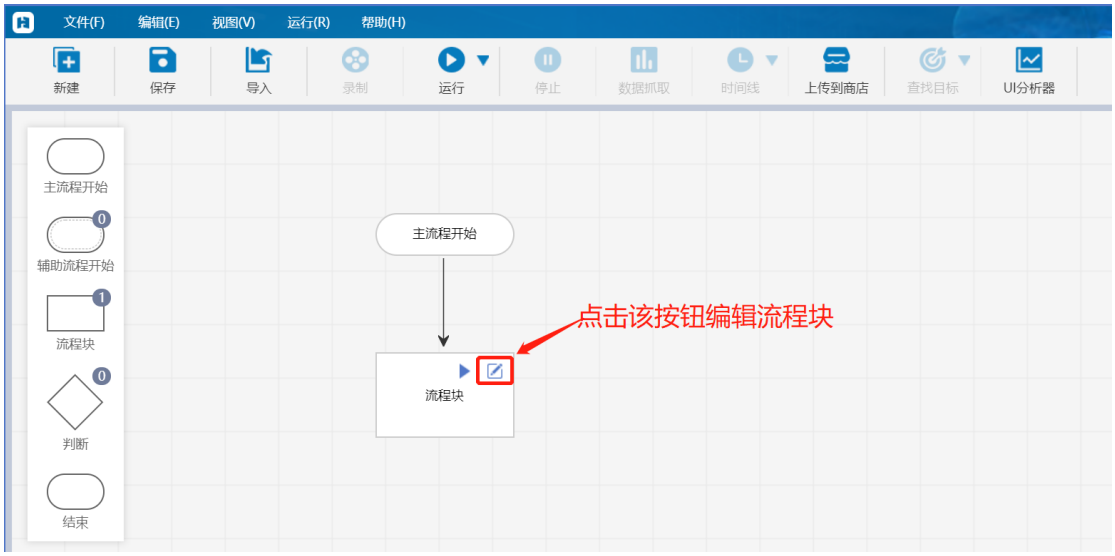
因为 Hibot 中“流程图”的初衷，是为了让设计 RPA 流程的“业务专家”和使用 RPA 的“一般工作人员”能够更好的沟通。双方在设计初期就确定大致步骤，划分流程块，然后，业务专家再负责填写每个流程块里面的细节，而一般工作人员就无需关注这些细节了。显然，在这个阶段，如果流程块的数量过多，沟通起来自然也会更加困难。

在 Hibot 的工具栏上，有一个“运行”按钮。在流程图界面中，按下这个按钮以后，会从“开始”组件开始，依次运行流程中的各个组件。而每个流程块上还有一个蓝色小三角形，实际上也是一个按钮，按下之后，就会只运行当前的流程块。这个功能方便我们在开发 RPA 流程时，把每个流程块拿出来单独测试。

每个流程块上还有一个形状类似于“纸和笔”的按钮，按下之后，可以查看和编写这个流程块里面的具体内容。具体的编写方法，通过“可视化视图”来完成。

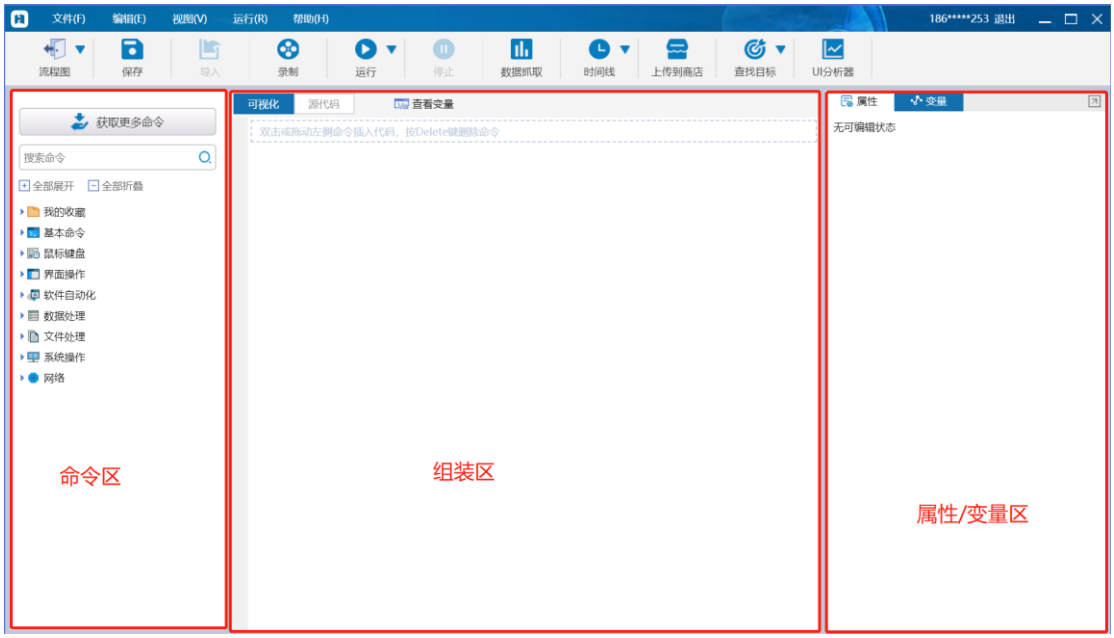
## 1.3 可视化视图

每个流程块上都有一个形状类似于“纸和笔”的按钮，点击该按钮，可以查看和编写这个流程块里面的具体内容，界面从“流程视图”转到“可视化视图”。




流程图中点击编辑流程块

Hibot 编写流程块的“可视化视图”，界面如下：




流程块编辑界面(可视化视图)

图中用三个红框标明了三个主要区域，从左到右分别是命令区、组装区、属性/变量区。

所谓命令，是指在一个流程块当中，需要告知 Hibot 具体每一步该做什么动作、如何去做。Hibot 会遵循我们给出的一条条命令，去忠实的执行。继续前面的例子，假如流程块是“把冰箱门打开”，那么具体的命令可能是：

- 找到冰箱门把手
- 抓住冰箱门把手
- 拉开冰箱门

当然，和前面一样，这个例子只是打个比方，Hibot 并不能把冰箱门打开。Hibot 所能完成的几乎所有命令，都分门别类地列在左侧的“命令区”，也就是上图中的第一个红框。包括模拟鼠标、键盘操作，对窗口、浏览器操作等等多个类别，每个类别包含的具体的命令还可以进一步展开查看。

图中第二个红框所包含的区域，称之为“组装区”，我们可以把命令在这里进行排列组合，形成流程块的具体内容。可以从左侧的“命令区”，双击鼠标左键或者直接拖动，把命令添加到“组装区”，也可以在组装区拖动命令，调整它们的先后顺序，或者包含关系。具体的操作方式参见相关实验教程。

命令是我们要求 Hibot 做的一个动作，但只有命令还不够，还需要给这个动作加上一些细节，这些细节就是我们要引入的第四个概念：属性。如果说命令只是一个动词的话，那么属性就是和这个动词相关的名词、副词等，它们组合在一起，Hibot 才知道具体如何做这个动作。

还用上面的例子来说，对于命令“拉开冰箱门”，它的属性包括：

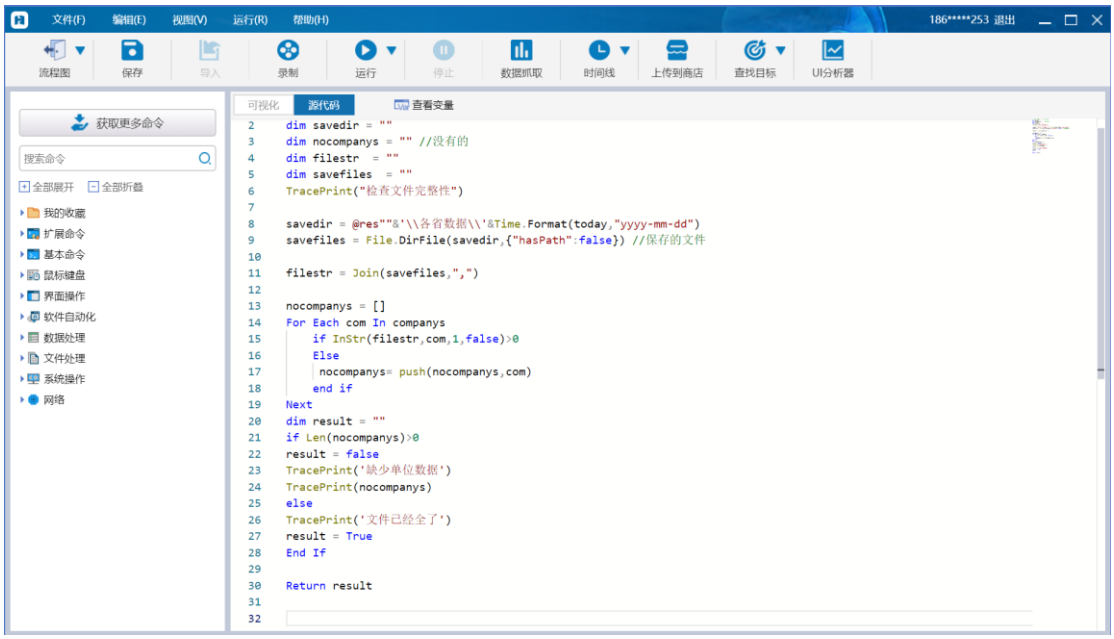
- 用多大力气
- 用左手还是右手
- 拉开多大角度

在编写流程块的时候，只需要在“组装区”用鼠标左键单击某条命令，将其置为高亮状态，右边的属性变量区即可显示当前命令的属性，属性包含“必选”和“可选”两大类。一般来说，Hibot 会为您自动设置每一个属性的默认值，但“必选”的属性还是请关注一下，可能经常需要您根据需要进行修改。对于“可选”的属性，一般保持默认值就好，只有特殊需求的时候才要修改。

您目前看到的组装区的展示方式，称为“可视化视图”。在这种视图中，所有命令的顺序、包含关系都以方块堆叠的形式展现，且适当的隐藏了其中的部分细节，比较容易理解。“可视化视图”体现出 Hibot 作为 RPA 平台的“简单”这一重要特点，为此，Hibot 的设计者们在“可视化视图”的表现方式、详略程度、美观程度方面都有过认真的思考和碰撞，达到了相对比较均衡的状态。即使是没有任何编程经验的新手，看到“可视化视图”，也可以大致掌握其中的逻辑。

## 1.4 源代码视图

在组装区的上面，有一个可以左右拨动的开关，左右两边的选项分别是“可视化”和“源代码”，默认是在“可视化”状态。我们可以将其切换到“源代码”状态，属性变量区会消失，组装区会变成如下图所示的样子：




流程块编辑界面(源代码视图)

采用这种方式展现的组装区，称为“源代码视图”。与“可视化视图”类似，“源代码视图”实际上也展现了当前流程块中所包含的命令，以及每条命令的属性。但没有方块把每个命令标识出来了，也没有属性区把每个属性整齐的罗列出来了，而是全部以程序代码的形式来展现。

如果您对 Hibot 已经比较熟悉了，那么切换到源代码视图，手不离开键盘即可书写命令和属性。Hibot 对源代码视图进行了很多体验上的优化，能帮您快速选择所需的命令，快速填写各个属性。


在源代码视图中添加命令

可视化视图和源代码视图描述的都是同一个流程块，它们实际上是同一事物的两种不同展现方式，其内涵都是一模一样的。可视化视图以图形化的方式，突出了各个命令，以及它们之间的关系，适合展现流程块的整体逻辑；源代码视图以程序代码的方式，突出了流程块的本质，并充分展现了其中的所有细节。

飞机的两种视图

无论使用哪种视图，都可以随时切换到另一种视图。无论在一种视图上编写了什么内容，切换到另一种视图以后，这些内容都会 100%保留，并以另一种视图的形式展现出来，反之亦然。另外，源代码视图还有一个好处，当您在论坛上、QQ 群里向其他人求助的时候，只要切换到源代码视图，把源代码复制粘贴，即可以文本的方式展现您的流程块。对方可以直接阅读源代码，也可以把源代码的文本粘贴到自己的 Hibot 中，并切换到可视化视图查看。这样交流的效率会大大提高。

在源代码视图中使用的编程语言，是 Hibot 自研的 BotScript 语言，具体的语言特性，将在后文详细描述。

## 2. 目标选取

Hibot 提供了一种全自动的选取目标的方式，我们以“鼠标”类别中的“点击目标”命令为例来说明。

假设我们要执行一个最简单的流程，这个流程只有一个步骤：点击 Windows 的开始菜单按钮（默认位置在左下角）。首先，新建一个流程，然后，打开其中唯一的流程块，接着，在“可视化”视图中找到“点击目标”命令，用拖动或者双击的方式将其插入组装区。

在组装区中，现在已经有一条命令了。我们会注意到，这条命令上有一个按钮，文字是“查找目标”，还有一个瞄准器样子的图标。如下图所示：


“点击目标”命令和“查找目标”按钮

点击这个按钮，Hibot 的界面暂时隐藏起来了，出现了一个红边蓝底的半透明遮罩，我们称之为“目标选择器”。鼠标移动到什么地方，这个目标选择器就出现在什么地方，直到我们单击鼠标左键，目标选择器消失，Hibot 的界面重新出现。在鼠标按下的时候，目标选择器所遮住的界面元素，就是我们选择的目标。

在按下鼠标之前，请先耐心移动鼠标，直到目标选择器不多不少的恰好遮住了您要操作的界面元素为止。

我们可以试一下，用目标选择器遮住开始菜单按钮，注意是恰好遮住，不多不少。当遮罩变成了下图所示的状态时，再单击左键完成选择。当然，下图是在 Windows 10 操作系统中的样子，对于其他版本的 Windows 操作系统，样子可能会有区别，但原理不变。

用“目标选择器”选中开始菜单按钮

一旦选中之后，Hibot 的界面重新出现，刚才按下的“查找目标”按钮也变成了目标界面元素的缩略图，这个缩略图仅供参考，帮助您记得刚才选中的是哪个目标，而不会对流程的运行有任何的影响。而且，这个缩略图实际上还是一个按钮，按下去以后，作用和刚才的“查找目标”一模一样。如果前面选择的目标不合适，或者不小心选错了，按这个按钮重来一次就好。

对于有目标的命令，在命令的属性中，有一条属性被称为“目标”。当我们还没有选择目标的时候，这个属性的值是一对花括号 {}，如下图左（此时实际上没有选择目标，所以如果运行的话，是一定会出错的）。而当我们选择了目标以后，这个属性的值会比较长，但仍然是被一对花括号所包围起来的，如下图右。


“目标”属性的值

不妨把这个长长的值粘贴到这里，它完整的样子其实是：

```
{"wnd":[{"app":"explorer","cls":"Shell_TrayWnd"},{"cls":"Start","title":"开始"}]}
```

当我们在后文中学习完 Hibot 的编程语言 BotScript 以后，就会知道这一长串内容实际上是 BotScript 中的一个“字典”数据类型。当然，现在并不需要掌握这些细节，只要知道这是一段特殊的数据即可。Hibot 在运行流程的时候，根据这段数据，就可以寻找我们指定的界面元素了。

如果您有过 Windows 的应用开发经验（如果没有，也没关系，这一段可以跳过去，不影响后续阅读），就会知道 Windows 上的应用程序实际上有很多开发框架，包括 SDK、MFC、WTL、WinForm、WPF、QT、Java 等等，如果再算上运行在 IE 和 Chrome 浏览器中的 Web 应用，类型就更多了。这些应用程序其实都提供了界面元素的查找、操作接口，从技术上来说，Hibot 无非就是调用这些接口而已。但是，这些接口的调用方法各不相同，甚至差异很大，即使是 IT 专家，也很难在短时间内对所有这些接口都驾轻就熟，更不用说一般用户了。

但如果用 Hibot，它们都是一样的“界面元素”，对它们进行查找和操作没有任何差异。比如，MFC 程序中可能有一个按钮，Chrome 浏览器中可能也有一个按钮，看起来都是按钮，但对这两个按钮分别模拟点击，技术上的差异几乎可以说是天壤之别。而在 Hibot 中，您完全无需关心这些区别，Hibot 已经把这些差异帮我们抹平了。从而实现了“强大”、“简单”、“快捷”三个指标的统一及平衡。

## 3. 目标编辑

在上一节中，我们看到 Hibot 的目标选择器是自动工作的。只要我们把鼠标移动到希望作为目标的界面元素上，遮罩会恰好遮住这个界面元素，并且会生成一段数据，Hibot 在运行的时候，用这段数据即可找到目标。

当然，凡是自动工作，都难免会出错。在使用目标选择器的时候，常见的问题是：

无论如何移动鼠标，都无法使遮罩恰好遮住要作为目标的界面元素（通常是遮罩太大，遮住了整个窗口）

遮罩可以恰好遮住界面元素，但用生成的数据去查找目标时，发生了如下情况：

- 错选：能找到界面元素，但找到的界面元素不是我们当初选取的
- 漏选：我们当初选取的界面元素明明存在，却找不到了

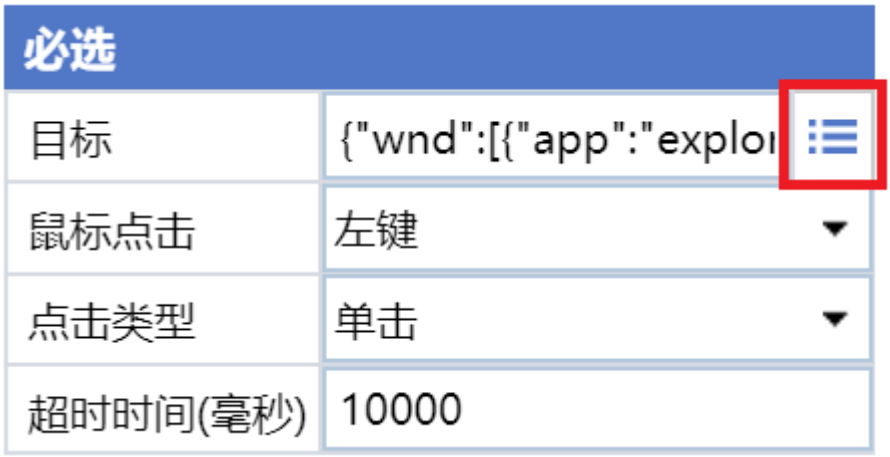
对于第一种情况，也就是无法遮住目标的情况，我们会在下一章用比较多的篇幅详细叙述。这里主要讨论的是第二种情况，也就是明明可以遮住目标，但在运行的时候，却发生错选或漏选的情况。

我们在上一节中提到，当选取目标时，Hibot 会生成类似于这样的一串数据，用来描述目标：

```
{"wnd":[{"app":"explorer","cls":"Shell_TrayWnd"},{"cls":"Start","title":"开始"}]}
```

Hibot 在运行流程的时候，就是根据这串数据中的描述，来查找目标的。所以，当发生错选或者漏选的时候，实际上就是这串数据出现了问题，需要对它进行修改。

如何修改呢？我们首先在“属性”栏，找到“目标”属性，这串数据就显示在后面的输入框里。既然是输入框，理论上可以直接编辑其内容，但输入框太小，编辑起来非常困难。需要修改目标的时候，推荐按输入框右边的按钮，如图中红框所示：




修改目标

按下这个按钮，会弹出一个“目标编辑器”的窗口。上半部分是缩略图，表示作为目标的界面元素的大致样子，Hibot 在查找目标的时候，并不会使用这张图片，仅仅是让您查阅参考的。下半部分是“控件筛选器”，把描述目标的那串数据用一个树形结构重新展示出来了，如图所示：


目标编辑器

实际上，这个树形结构里面，保存的是界面元素的某些特征，每一项都是一个特征，这些特征是 Hibot 自动选取的，只有当这些特征全部满足的时候，才会认为找到了界面元素。而且，由于界面元素是相互嵌套的，Hibot 不仅会记录作为目标的界面元素的特征，还会保存它的上面若干级的界面元素的特征。每一级的特征都必须全部满足才行。

以上图中 Windows 的开始菜单按钮为例，其中 0: Object 那一行及其下面的内容，代表的是开始菜单按钮的上一级界面元素（实际上是 Windows 任务栏）的特征，而 1: Object 那一行及其下面的内容，代表的才是开始菜单按钮本身的特征。在流程运行的时候，Hibot 会逐级查找，首先找到第一级的 Windows 任务栏，然后再在任务栏里面，找所有特征全部满足的开始菜单按钮。

这样严格的特征匹配，显然很容易造成“漏选”。比如，我们可以看到，Windows 为开始菜单按钮设定了一个“标题”，也就是 title 那一行，其内容是“开始”（这个标题通常不会让用户看到，但却是实际存在的）。Hibot 会把这个标题作为特征的一部分，因为一般来说，按钮的标题是不会变的。但是，如果有一天，这个按钮的标题发生了变化，就造成了漏选。

那么，该怎么修改呢？我们看到，在 title 前面有一个勾选框，默认是处于已勾选状态的。只需要点一下这个勾选框，将其置为未勾选的状态，Hibot 在找界面元素的时候，就不会再使用这个特征，即使标题发生了变化，也能找到。

但是，如果去掉了太多的特征，漏选是不太容易发生了，却会发生错选。举个极端一些的例子：如果把 cls: "Start"和 title: "开始"这两行全都取消勾选，显然，0: Object 所代表的 Windows 任务栏下面的任何一个界面元素，就都可以满足条件了，这就造成了错选。

所以，如果界面元素比较复杂，或者特征经常发生变化，如何准确的编辑目标，既不发生错选也不发生漏选，还是需要一定技巧的。很遗憾，这方面并没有

特定的规则，只能多多尝试，积累经验。下面有几条公共的经验，请读者先记住，然后再在实践中总结自己的经验。

- 有的特征名称您可能暂时不理解，比如 cls、aaname 等，可以暂时不管它们；
- 善用通配符 *，这个通配符代表“匹配任意内容”。比如有一个界面元素，其 title 特征的值是“姓名：张三”，后面的“张三”可能会变，但前面的“姓名：”不变。所以，可以用 title:"姓名：*"来作为特征，而不是把这条特征去掉；
- 去掉特征的时候要慎重。因为去掉特征虽然可以减少漏选，但会增加错选。在流程运行的时候，漏选一般比较容易发现，但错选未必能马上发现。

关于最后一条，值得特别说明一下：Hibot 在运行一个流程的时候，大多数的“有目标”命令在找不到目标的时候，都会抛出一个异常（除非是“判断目标是否存在”这样的命令），流程会马上停下来，并且报错（除非您使用了 Try…Catch 来捕获异常，具体用法请参考后文）。所以比较容易发现。而当发生错选的时候，Hibot 并不知道，还会继续往下运行，错误就不太容易发现了。

最后，需要提醒的是：同样的界面元素，在不同的操作系统、不同的浏览器上，可能特征也会发生变化。特别是 IE 和 Chrome 浏览器，在显示同一个页面的时候，同样的界面元素可能会有完全不同的特征。如下图所示，同样用 IE 和 Chrome 打开百度的首页，并且把百度的搜索框作为目标来选取，其特征具有较大的差异。


用 IE 和 Chrome，同一目标的特征不同

所以，在用 Hibot 制作流程，并且在别人的计算机上使用的时候，请尽量保持开发环境和生产环境的一致性，以减少不必要的错误。

## 4. 软件自动化

在 RPA 流程中，我们经常需要对 Excel、Word 等办公软件，或者浏览器等常用软件进行自动化操作。当然，这些软件都是有界面，也可以得到界面元素。理论上学习了有目标命令这一章，就可以对这些软件进行自动化操作了，但这样做起来会比较繁琐。因此，Hibot 特地把 Excel、Word、Outlook、浏览器、数据库

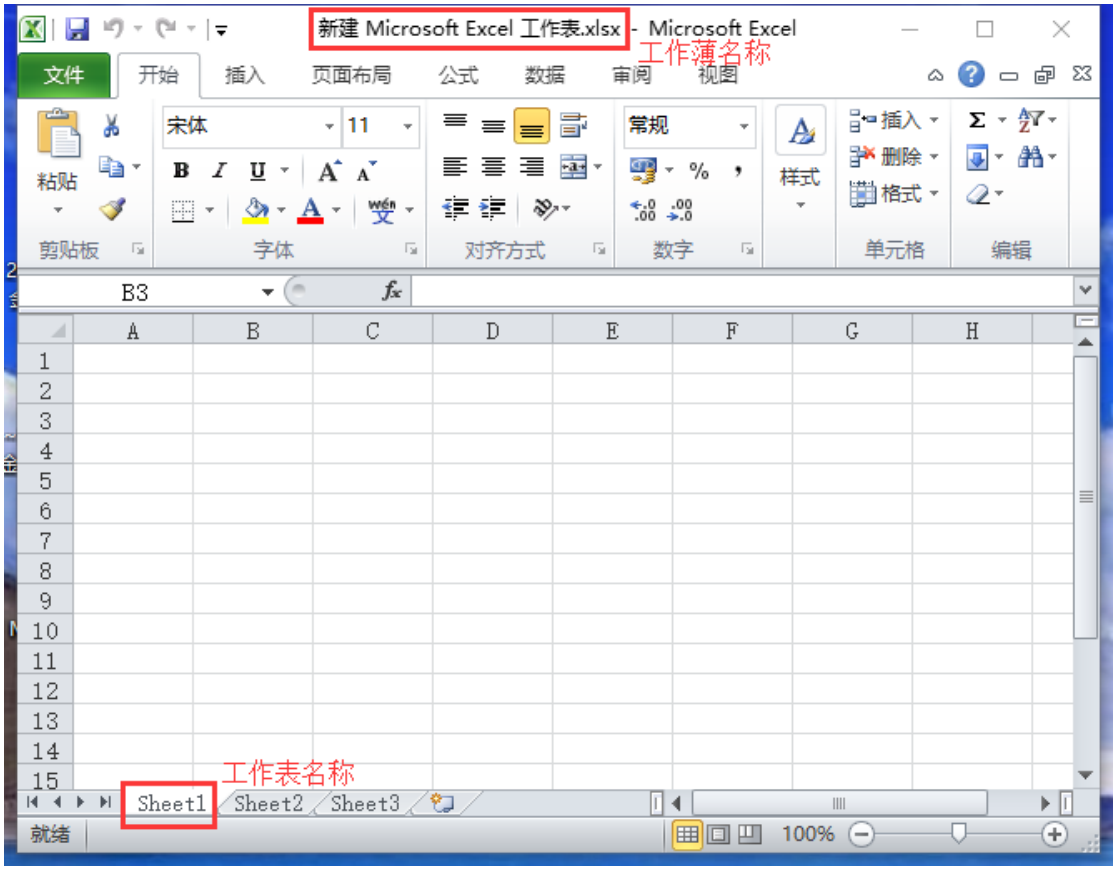
等软件的自动化操作封装成为专门的命令，通过这些命令来操作，会比界面上的模拟更高效、更方便。比如，虽然我们可以通过界面模拟来模拟真人的操作，打开、读写一个 Excel 文档，但是这样非常麻烦，而通过 Excel 自动化的命令，只需要一条命令就可以做到。

用 Hibot 自动化操作这些软件之前，您的计算机需要安装相应的软件。对于 Excel、Word 自动化，需要安装 Office 2007 以上版本，或者 WPS 2016 以上版本；对于浏览器自动化，需要安装 Internet Explorer（IE）、Google Chrome 或者火狐浏览器。

## 4.1 Excel 自动化

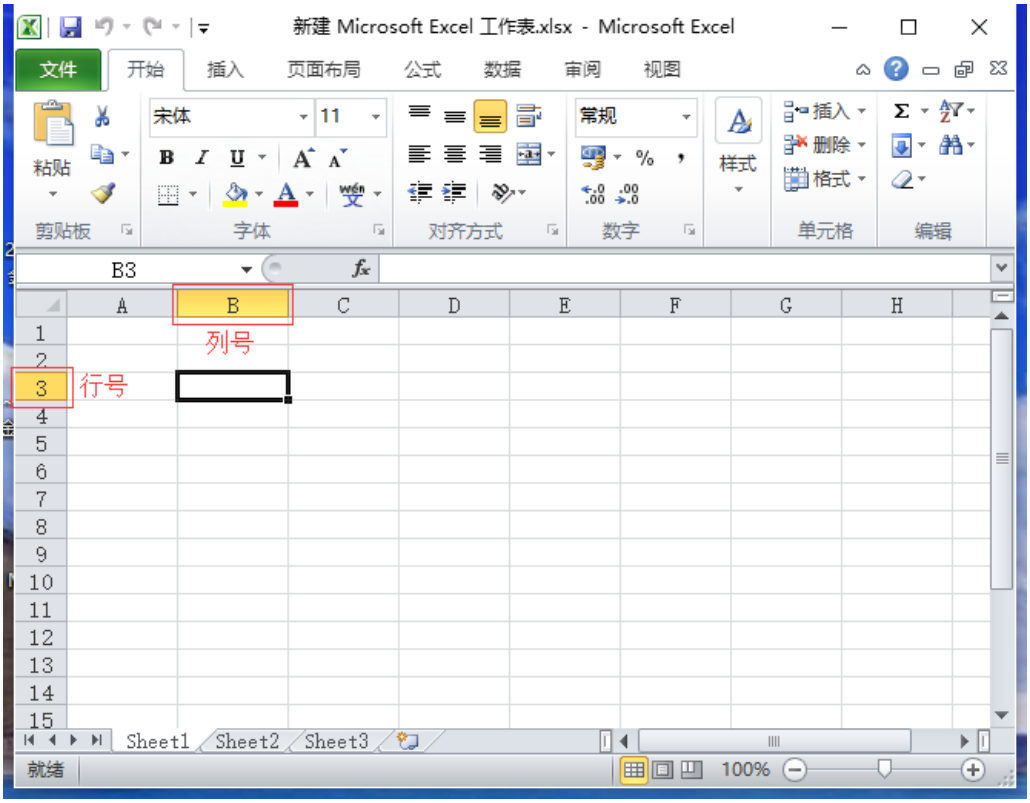
Excel 是 Office 办公软件的重要组成成员，它具有强大的计算、分析和图表功能，也是最常用、最流行的电子表格处理软件之一。对 Excel 实现自动化，是 RPA 流程中经常遇到的场景。

在实现 Excel 自动化之前，我们先明确几个概念：工作簿和工作表。工作簿是处理和存储数据的文件，一个 Excel 文件对应一个工作薄，Excel 软件标题栏上显示的是当前工作簿的名字。工作表是指工作簿中的一张表格。每个工作簿默认包含三张工作表，分别叫 Sheet1、Sheet2、Sheet3，当然也可以删除或者新增工作表，就是说工作薄和工作表是一对多的关系。




Excel 工作薄和工作表

Excel 中的工作表是一个二维表格，其中包含很多单元格，使用行号和列号可以确定一个单元格的具体位置，行号通常用 1,2,3,4……这样的数字序列表示；列号通常用 A,B,C,D……这样的字母序列表示。这样就可以用　列号+行号　来表示一个单元格，比如 B3 单元格，就是指第 3 行第 2 列交界位置的那个单元格。

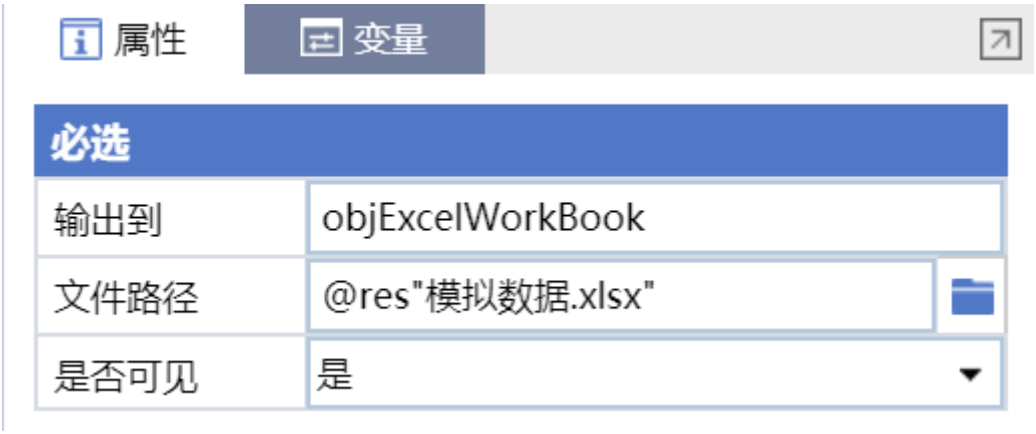



Excel 的行和列

用 Hibot 自动化操作 Excel 表格的时候，首先需要打开工作簿，后面的对工作表或单元格的各种操作，都是针对某个已经打开的工作簿进行的。另外，当自动化操作 Excel 表格结束以后，还需要关闭已经打开的工作簿。

我们来尝试用 Hibot 打开一个工作簿。在 Hibot Designer 的命令列表中，选中“软件自动化”并展开，再选中“Excel”并打开，排在第一位的就是“打开 Excel”命令，用这条命令可以打开一个 Excel 工作簿。

这条命令有三个属性，如下图所示。我们先看“文件路径”属性，这里需要指定一个 Excel 工作簿文件的路径，文件可以是 xls、xlsx、xlsm 等格式。前面说过，这个路径可以是绝对路径，也可以用诸如@res"模拟数据.xlsx"的格式来指代一个相对路径下的文件，相对的是您的流程所在的文件夹中，名为 res 的文件夹。另外，请注意在 Hibot 中，当字符串里出现\符号时，应写为\\。



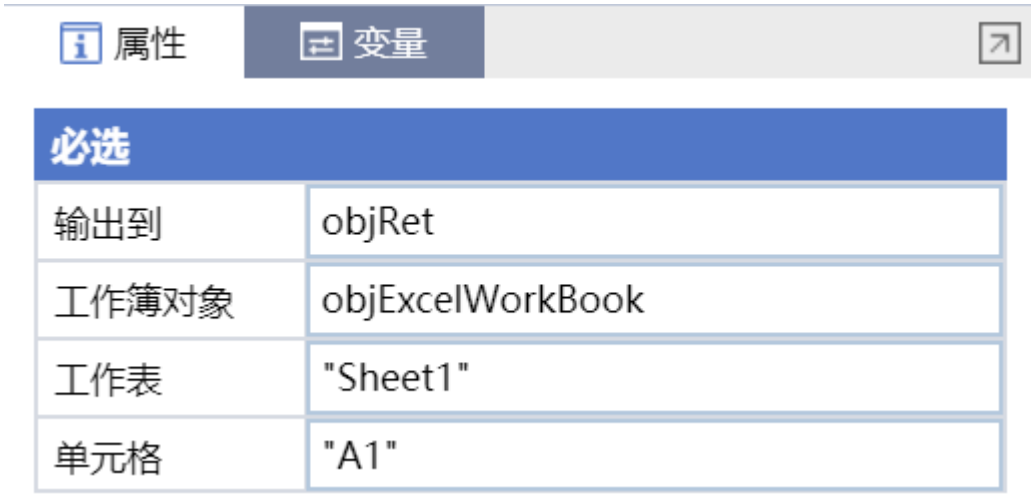

打开 Excel 工作簿

如果我们指定的工作簿文件存在，在流程运行的时候，会对这个文件进行操作。如果文件不存在，在流程运行的时候，会自动创建一个空白的 Excel 工作簿文件，并对这个新创建的文件进行操作。

下一个属性是“是否可见”，这是一个布尔类型的属性，其值只能是“是（True）”或者“否（False）”。当选择“是”的时候，这条命令会打开 Excel 软件，并且把这个工作簿显示出来。否则，可以在不显示 Excel 软件界面的情况下，仍然正常读取或修改这个工作簿文件的内容。

还有一条属性是“输出到”，这里必须填写一个变量名，这个变量指代了我们打开的 Excel 工作簿，我们称之为一个“工作簿对象”。后面在对工作簿进行各种读取、修改操作的时候，仍然需要把这个变量填入到相应命令的“工作簿对象”属性中，表明操作是针对这个工作簿进行的。比如，上图中我们在打开工作簿的时候，“输出到”变量是 objExcelWorkBook，后续的 Excel 操作命令，其“工作簿对象”属性都需要填写 objExcelWorkBook。

我们来尝试读取这个工作簿的 Sheet1 工作表里面的 A1 单元格的内容。插入一条“读取单元格”命令，我们可以看到这条命令的属性如下图所示：




读取单元格

如上所述，这里的“工作簿对象”属性，应该和“打开 Excel 命令”的“输出到”属性是一样的，所以我们需要填写 objExcelWorkBook，表明我们是从刚才打开的工作簿中读取单元格内容。

“工作表”和“单元格”属性都采用字符串的形式（需要加双引号表示这是一个字符串），按照 Excel 的习惯来填写，这里我们分别填写"Sheet1"和"A1"。

“输出到”属性中还需要填写一个变量名，表示把读取到的单元格内容输出到这个变量中。如果单元格的内容是数值，那么这个变量的值也会是一个数值；如果单元格的内容是字符串，那么变量的值自然也是字符串。

在我们的工作中，经常需要读取 Excel 工作簿的多个单元格里面的数据，如果用 Hibot 每次读取一个单元格，既低效又麻烦。实际上，强大的 Hibot 提供了“读取区域”的命令，可以一次性把一个矩形范围内所有单元格的内容全部读取出来。我们试着插入一条“读取区域”命令，它的属性如下图所示。


读取区域

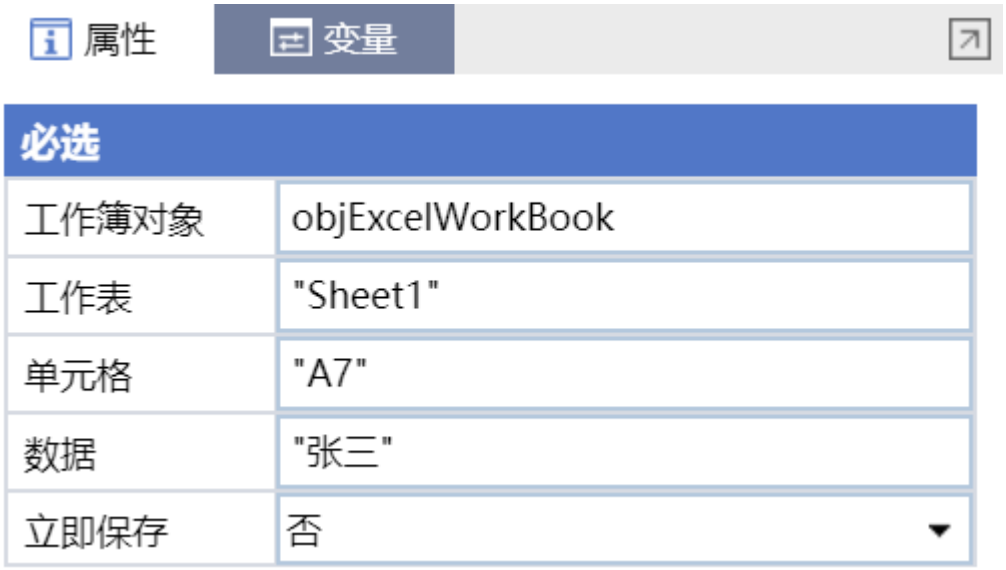
从上图可以看出，“读取区域”命令与“读取单元格”命令相比，有两个属性完全一致，即“工作薄对象”和“工作表”，这两个属性表示需要读取哪个工作薄的哪个工作表的内容。

“区域”属性同样采用字符串的形式（需要加双引号表示这是一个字符串），同样按照 Excel 的习惯来填写，这里填写的是"A2:B6"，表示读取的是从左上角 A2 单元格到右下角 B6 单元格，共计 6 行 2 列 12 条数据。

“输出到”属性中填写了一个变量名 arrayRet，读取到的内容将会输出到这个变量中。我们在“读取区域”命令之后，加入一条“输出调试信息”语句，将 arrayRet 这个变量的值打印出来。从输出信息可以看到，“读取区域”命令输出的是一个二维数组，例如：[ [ "马花花", 123456 ], [ "刘弱西", 654321 ], [ "王兼邻", 987654 ], [ "马雨", 741258 ], [ "公孙永耗", 951753 ] ]

现在我们还不知道“数组”、“二维数组”是什么，这些概念会在后文中详细讲解，现在我们只需要知道：利用 Excel 的“读取区域”命令，可以将一个 Excel 表格某块区域内的数据全部读取出来，并放到了一个变量 arrayRet 中。

既然能读取，同样也能够写入。Hibot 同样提供了一系列的 Excel 写入命令来修改工作薄的内容。我们来尝试将上述工作簿的 Sheet1 工作表里面的 A7 单元格的内容写为“张三”。在“打开 Excel”命令之后插入一条“写入单元格”命令，我们可以看到这条命令的属性如下图所示：




写入单元格

其中，“工作簿对象”、“工作表”和“单元格”三个属性的含义与“读取单元格”命令一致，表示本条命令操作的是哪个“工作簿对象”的哪个“工作表”的哪个“单元格”。

“数据”属性中填入的是即将写入单元格的数据，可以是数字常量、字符串常量，也可以是变量或者表达式。

Excel 写入类命令，还有一个很重要的属性 —— “立即保存”属性，如果这个属性选择“是”，那么写入操作会被立即保存，就好比我们手动修改 Excel 文件内容后，立即按“ctrl+S”进行保存一样；而如果这个属性选择“否”，那么写入操作将不会被立即保存，除非单独调用一次“保存 Excel”命令，或者在“关闭 Excel”命令的“立即保存”属性选择“是”，两种方法效果一样，都可以保存 Excel 修改的内容。

其它的 Excel 写入类命令的用法与“写入单元格”命令类似，在此不再赘述。需要注意的是：每个写入类命令的“数据”属性，必须与这条写入命令的写入范围一致，这样才能保证数据能够正确写入。就是说，写入一个单元格，“数据”属性就应该是一个单元格的数据；写入一行，“数据”属性就应该是一行单元格的数据（一维数组），且数组的长度与该工作表数据的列数相等；写入区域，“数

据”属性就应该是几行几列单元格的数据（二维数组）。如果不一致，很容易报错或者出现写入 Excel 数据错位的情况。

## 4.2 Word 自动化

与 Excel 类似，Word 也是 Office 办公软件的重要组成成员。Word 格式的文档几乎是办公文档的事实标准，对 Word 实现自动化，也是 RPA 流程中不可回避的一环。

同样地，用 Hibot 自动化操作 Word 文档的时候，首先需要打开这个 Word 文档，后面对文档内容的各种操作，都是针对这个已经打开的文档进行的。当操作 Word 文档结束以后，还需要关闭已经打开的文档。

我们来尝试用 Hibot 打开一个 Word 文档。在 Hibot Designer 的命令列表中，选中“软件自动化”并展开，再选中“Word”并打开，排在第一位的就是“打开文档”命令，用这条命令可以打开一个 Word 文档。

这条命令有五个属性，如下图所示。我们先看“文件路径”属性，这里需要指定一个 Word 文件的路径，文件可以是 doc、docx 等格式，其它注意事项与上一节的“打开 Excel”命令的“文件路径”属性一致。这里我们打开的是 res 目录下的 模拟文档.docx 文件。

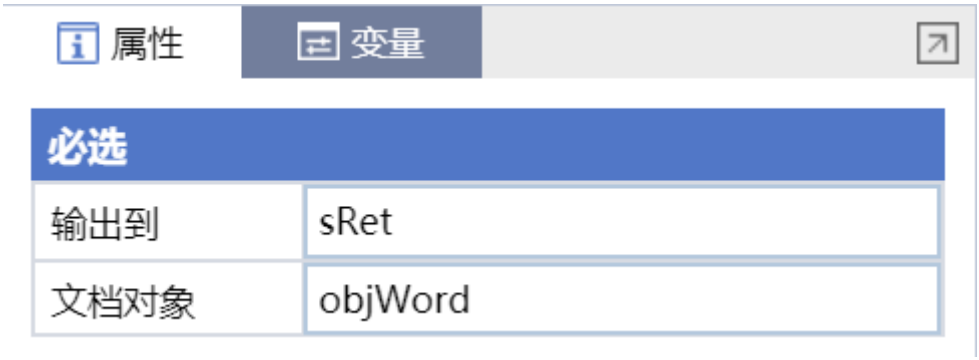

打开 Word 文档

接下来是“访问时密码”和“编辑时密码”两个属性，这是什么意思呢？有时候，出于隐私的考虑，我们的文档不希望他人能够打开，或者打开后不能修改，因此就给 Word 文档设置密码，密码分为两个：一个叫“访问密码”，输入正确的访问密码就可以打开这个文档；一个叫“编辑密码”，输入正确的编辑密码就可以修改这个文档。这里的“访问时密码”和“编辑时密码”两个属性就是用来自动化访问带密码的 Word 文档的。如果所操作的 Word 文档没有设置密码，那么这两个属性保持为""不变即可。

“是否可见”属性与“打开 Excel”的“是否可见”属性含义相同，表示的是：在进行 Word 文档自动化操作时，是否显示 Word 软件界面。

还有最后一条“输出到”属性，与“打开 Excel”的“输出到”属性含义类似，这里必须填写一个变量名，这个变量指代了我们打开的 Word 文档，后面在对该文档进行各种读取、修改操作的时候，仍然需要把这个变量填入到相应命令的“文档对象”属性中，表明操作是针对这个打开的文档进行的。比如，上图中我们在打开文档的时候，“输出到”变量是 objWord，后续的 Word 操作命令，其“文档对象”属性都需要填写 objWord。

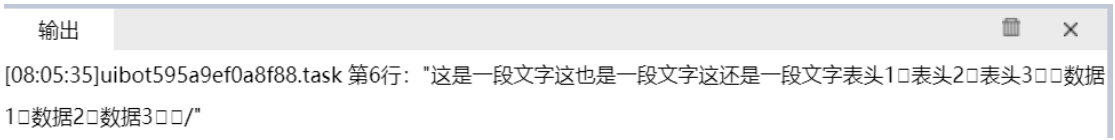
接下来，我们读取这个 Word 文档的内容。在“打开文档”命令之后，插入一条“读取文档”命令，这条命令的属性如下图所示：




读取 Word 文档

如上所述，“文档对象”属性和“打开文档”的“输出到”属性一致，都为 objWord，表明我们是从刚才打开的文档中读取内容。

“输出到”属性填写了一个变量名 sRet，表示把读取到的内容输出到变量 sRet 中。我们再添加一条“输出调试信息”命令，将 sRet 的内容打印出来，运行后，可以看到如下结果：




读取 Word 文档的输出结果

我们打开原始文档来对比一下，可以看到：原始 Word 文档包括文字、表格和图片，且文字带格式信息，“读取文档”命令会将文档中的文字内容全部读取出来，但是暂时不支持读取文字的格式、表格的状态和图片。

这是一段文字

这也是一段文字

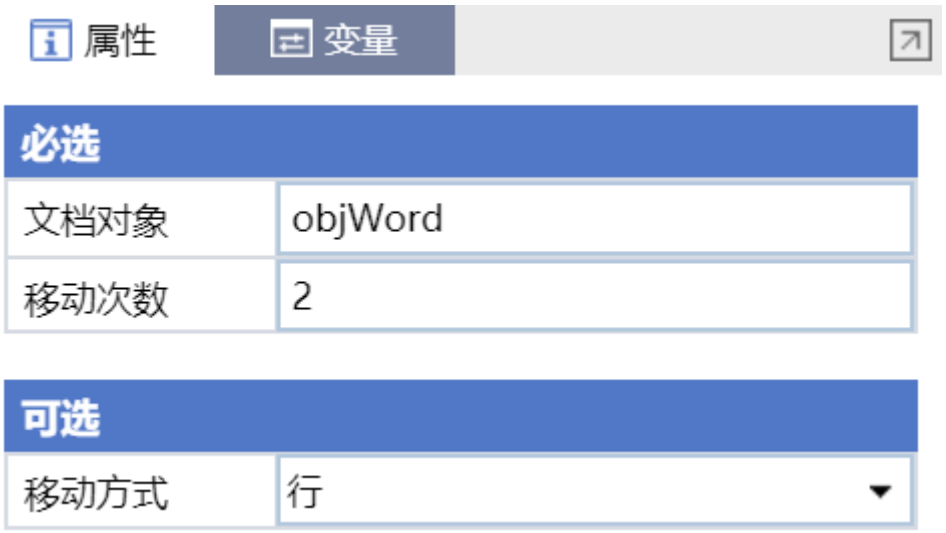
这还是一段文字

| 表头 1 | 表头 2 | 表头 3 |
| --- | --- | --- |
| 数据 1 | 数据 2 | 数据 3 |

原始 Word 文档

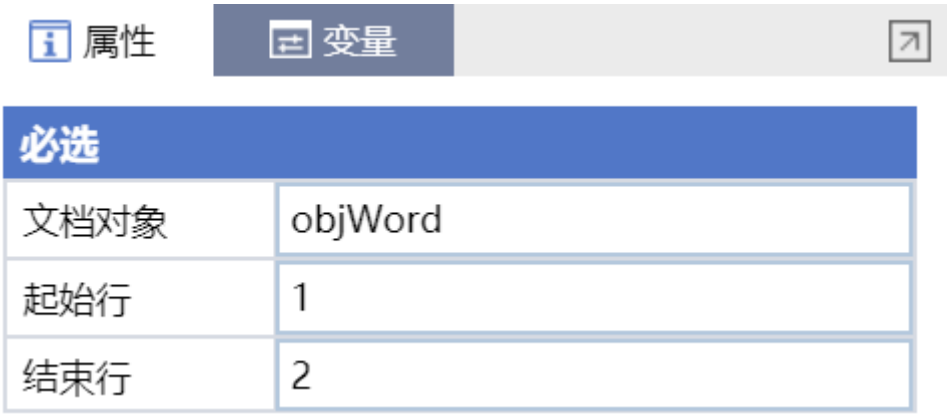
“读取文档”命令操作的是整个文档，类似命令还有“重写文档”、“保存文档”、“文档另存为”、“关闭文档”、“获取文档路径”等，这些命令都是对整个文档的操作。如果需要对文档进行更细粒度的操作，就需要涉及到 Word 中一个重要的概念：**焦点**。所谓焦点，指的是当前选中的区域，这块区域在 Word 中通常会高亮显示；如果没有选中区域，当前光标位置即为焦点。即：“焦点”=“选中”或“光标”。Word 的操作大都针对焦点进行，例如，要改变一段文字的字体，首先要选中这段文字，才能修改文字的大小、颜色、样式等；在 Word 中插入文字、图片等内容，也需要先将光标移动到插入点。

我们来看看 Hibot 中如何实现焦点的设置和切换。插入一条“设置光标位置”命令，这条命令可以将光标焦点设置到指定位置。这条命令有三个属性：“文档对象”属性，就是上文所述的文档对象 objWord；“移动次数”属性需要与可选属性中的“移动方式”属性配合使用，指的是光标按照“移动方式”移动多少次，“移动方式”属性有三个选项，分别是“字符”、“行”和“段落”，分别代表光标向右移动一个字符、向下移动一行和向下移动一个段落。在这里，“移动方式”设置为“行”，“移动次数”设置为 2，表示焦点设置为初始焦点下移两行，也就是第三行。需要注意的是：移动次数不能为负数，也就是说，光标不能向左移动、向上移动。




设置焦点

我们再插入一条“选择行”命令，这条命令可以选中特定的行。这条命令有三个属性：“文档对象”属性，就是上文所述的文档对象 objWord；“起始行”属性和“结束行”属性限定了选中的范围，在这里，“起始行”设置为 1，“结束行”设置为 2，表示选中第 1 行到第 2 行，一共 2 行。

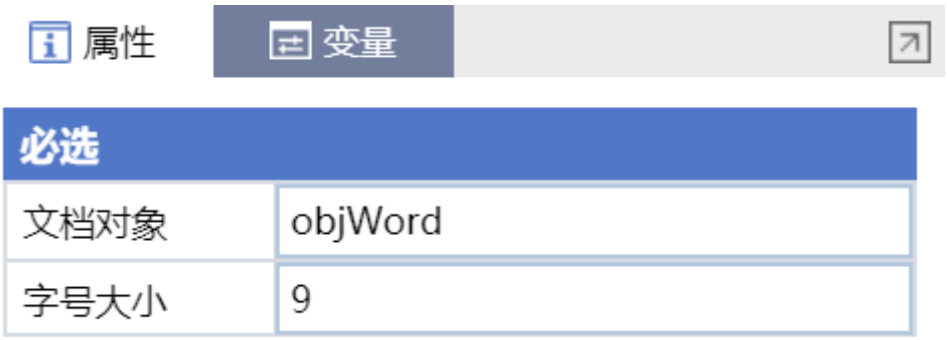



选择行

但是，在实际的应用中，单独使用“设置光标位置”命令和“选择行”命令进行光标焦点的设定，实际效果并不好，这是为什么呢？原来，Word 虽然是一个所见即所得的可视化图文混排软件，但是 Word 中同样存在一些看不见的格式标记，这些格式表示或多或少会影响 Word 文档中“字符”、“行”和“段落”的计算，导致焦点定位不准的问题。那如何来解决这个问题呢？这里教给大家一个小技巧：首先，我们在 Word 文档中，需要插入或者编辑的地方设置一个特殊的标记，例如插入名称字段，就设置在 Name；然后，使用“查找文本后设置光标位置”命令，这条命令有两个关键属性，一个是“文本内容”属性，填写前面的 Name 即可，一个是相对位置属性，选择“选中文本”，这样就可以找到 Name 这个标记并选中这个标记的内容；最后，使用“写入文字”命令将选中内容替换

成需要的内容。我们可以在 Word 文档中多设置几个这样的特殊标记，然后重复利用“查找文本后设置光标位置”命令，达到 Word 文档填写的目的。

继续前述内容，将光标移动到指定位置或者选中指定内容后，就可以执行具体的编辑操作了，包括插入内容、读取内容、删除内容、设置内容的格式、剪切/复制/粘帖等等，我们这里以“设置文字大小”命令为例。在“选择行”命令之后，插入一条“设置文字大小”命令。这条命令有两个属性：“文档对象”属性，就是上文所述的文档对象 objWord；“字号大小”属性指定选中文字的字号大小，在这里，“字号大小”设置为 9，表示选中文字的字号大小统一设置为 9。




设置文字大小

## 4.3 浏览器自动化

浏览器的自动化是软件自动化的一个重要组成部分，从特定上的网站上抓取数据、自动化操作 Web 形态的业务系统都需要基于浏览器进行自动化操作。

首先，我们需要打开一个浏览器，这个功能是通过“启动新的浏览器”命令来实现的。当然，如果计算机此时已经打开了一个浏览器，我们也可以直接利用这个打开的浏览器进行后续操作，此时，只需要一条“绑定浏览器”命令，其效果和“启动新的浏览器”命令是一样的。


启动新的浏览器

“启动新的浏览器”命令的属性如下：“浏览器类型”属性指定启动哪个浏览器。Hibot 目前支持 IE 浏览器、Google Chrome 浏览器、火狐浏览器、Hibot 自带浏览器四种浏览器，其中，前三种浏览器需要你的电脑提前安装好，Hibot 自带浏览器是 Hibot Designer 5.0 版本后自带的浏览器。在这里，我们选择的是“Hibot Brower”，即 Hibot 自带的浏览器。相比其它三种浏览器，Hibot 浏览器有如下优点：第一、无需安装任何浏览器扩展，即可选取目标元素（Google Chrome 和 Firefox 都需要安装扩展，在这个过程中，有时候会有一些意外的情况发生，例如被杀毒软件拦截等）；第二、Hibot 浏览器可以选取到跨域网页中的目标元素（使用其它浏览器登录网易、QQ 等邮箱时无法找到用户名和密码输入框）；第三、Hibot 浏览器可以直接调用所访问页面内的 JavaScript 方法。基于上述优点，我们推荐优先使用 Hibot 浏览器。当然，也有些比较特殊的网站，只能使用特定的浏览器才能正确打开和操作，比如某些国内银行网站，某些政府网站等，都只能使用 IE 浏览器才能正确打开和操作，这个时候，“浏览器类型”属性就只能选择“IE 浏览器”了。

“打开链接”属性表示打开浏览器时，同时打开哪个网址。在这里填写的是″www.baidu.com″，表示打开浏览器时，同时打开百度网站。当然，这里也可以暂时不填，后面再使用“打开网页”命令单独打开一个网址。

“超时时间”属性的意思是，如果出现异常情况，比如浏览器找不到，或者指定的链接打不开时，Hibot 会反复进行尝试，直到超过指定的时间，也就是“超时时间”。

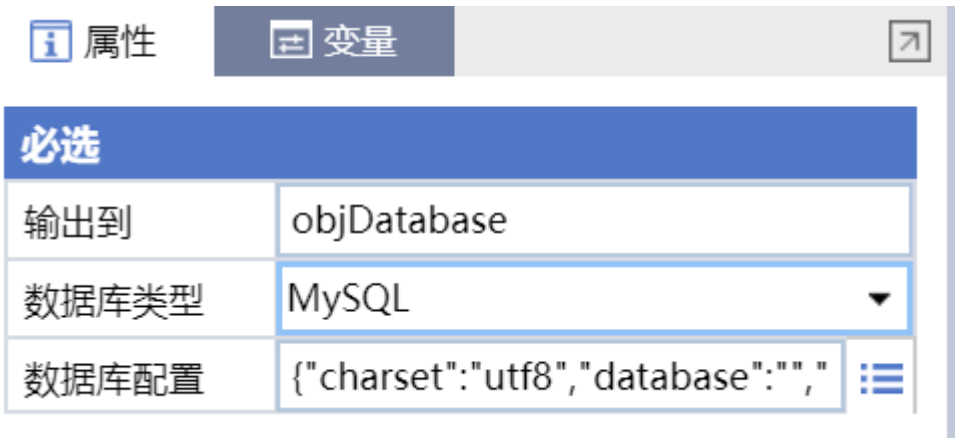
有两个可选属性也比较常用：一个是“浏览器路径”属性。有时候，我们会在同一台电脑上安装了两个不同版本的浏览器软件，这时，我们可以通过指定“浏览器路径”属性来打开某个特定版本的浏览器。如果不指定这个属性，系统会去浏览器默认安装目录下查找并启动浏览器软件；另一个是“浏览器参数”属性，我们知道，浏览器其实是非常强大的，浏览器除了能够默认启动外，还可以通过自定义启动参数，包括默认打开某些网页、展现方式（全屏等）、启用或禁用某些功能等，来启动一个个性化的浏览器。具体每种浏览器可以配置哪些启动参数，请参见相应的说明文档。

启动浏览器后，就可以针对浏览器及浏览器中显示的网页进行一系列的操作，我们可以浏览网页、在网页中输入文字、点击网页中的链接和按钮等。比如，打开了百度网站，我们可以在百度主页的输入框中，输入“Hibot”，并点击“百度一下”按钮，就可以得到“Hibot”在百度中的搜索结果。这些操作都可以通过前面章节的“有目标命令”完成，搜索结果也可以通过“数据处理”命令进行处理，完成数据抓取、数据分析等功能，这一块功能将在后续教程“数据处理”中详细讲解，这里就不再展开。

## 4.4 数据库自动化

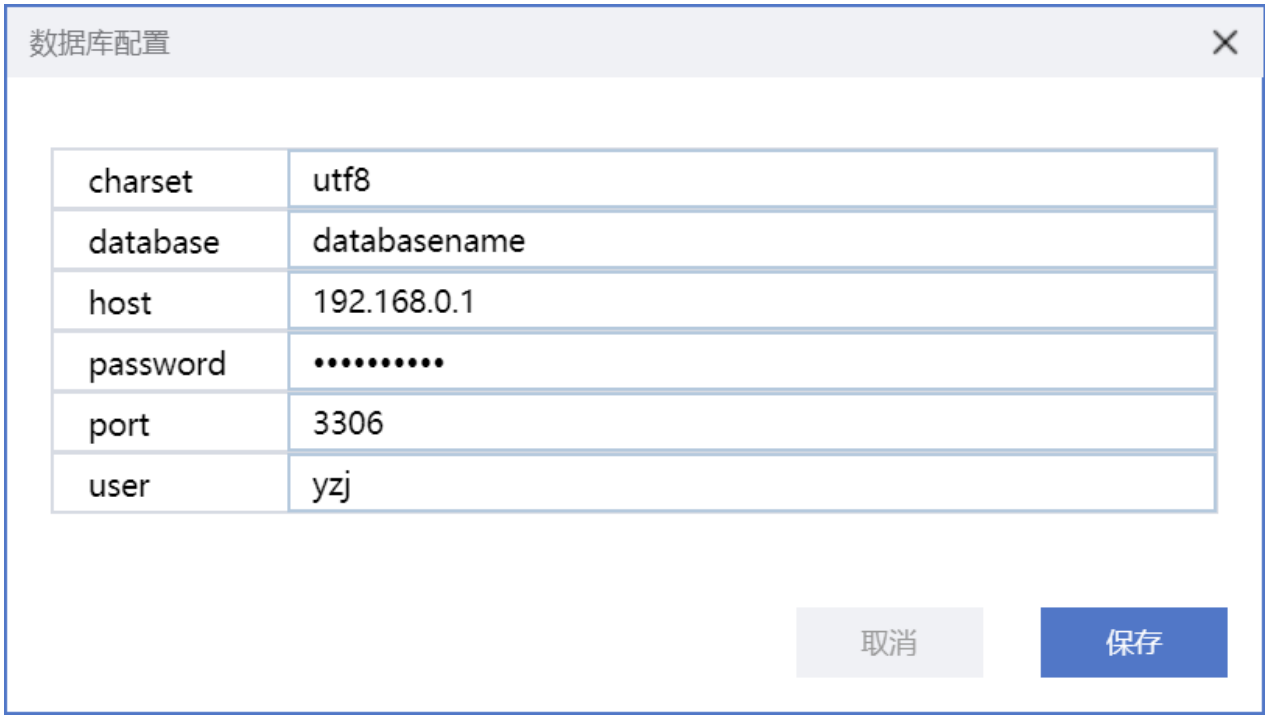
一个信息系统中，最重要的就是数据，现在，几乎所有的信息系统都将数据存储在数据库中。除了使用客户端访问数据库之外，有时候，也需要直接对数据库进行访问和操作，因此，针对数据库的自动化操作也成为了 RPA 中不可或缺的一环。所谓数据库自动化操作，指的是在保证数据安全的前提下，直接使用用户名和密码登录数据库，并使用 SQL 语句对数据库进行操作。关于 SQL 的基础知识，请参见网络上的 SQL 教程

我们来看一下具体如何操作数据库。首先，需要连接数据库。在“软件自动化”的“数据库”目录下，选择并插入一条“创建数据库对象”命令，该命令将创建一个连接指定数据库的数据库对象。




创建数据库对象

“创建数据库对象”命令有三个属性：“数据库类型”属性指定了创建的数据库对象的类型，Hibot 目前支持 MySQL、SQL Server、Oracle、Sqlite3 四种数据库类型。“数据库配置”属性是一个字符串，这个字符串描述了创建数据库对象时的一些关键信息。这个字符串比较长，也不太容易看懂，不过没关系，点开这个属性可以看到，“数据库配置”属性由一些子属性组成。




数据库配置

第一个“charset”，指的是数据库的字符集，我们保持默认“utf8”不变即可；第二个“database”，指的是我们连接的数据库的名称；“host”和“port”指的是数据库的地址和端口号，这里我填的是“192.168.0.1”和“3306”，表明数据库可以通过“http://192.168.0.1:3306”进行访问；“user”和“password”

指的是访问数据库的用户名和密码；通过配置上述几个参数，就可以成功创建数据库对象。

当然，每种类型的数据库，其参数可能不完全相同，比如 Oracle 数据库，没有“database”参数，只有“sid”参数，但是其含义是类似的。Sqlite3 数据库跟另外三个数据库差别比较大：MySQL、SQL Server、Oracle 是典型的关系型数据库，而 Sqlite3 是文件型数据库，因此 Sqlite3 的“数据库配置”属性，只有“filepath”一个子属性，指明了所操作的 Sqlite3 数据库文件的路径。

“输出到”属性，这个属性填写一个变量名，这个变量会保存创建的数据库对象，在这里我们填写 objDatabase，后续的所有数据库操作都针对 objDatabase 数据库对象进行。

成功创建数据库对象后，接下来可以对数据库进行操作了。Hibot 提供两种数据库操作：一种是查询数据，对应“执行单 SQL 查询”和“执行全 SQL 查询”两条命令；一种是对数据库、表和表中数据进行修改，对应“执行 SQL 语句”和“批量执行 SQL 语句”两条命令。

我们先来看看“执行单 SQL 查询”命令，这条命令可以执行一条 SQL 查询语句，并且返回查询到的第一条结果。插入一条“执行单 SQL 查询”命令，可以看到这条命令有三个属性：一个是“数据库对象”属性，这个属性填入刚刚得到的数据库对象 objDatabase；一个是“SQL 语句”属性，这个属性填入将要执行的查询语句，这里填入的是"select * from table1"，意思是查询 table1 表的所有数据，并返回第一条结果；第三个属性是“输出到”属性，这里填入一个变量 iRet，表示 SQL 语句的执行结果，我们通过判断 iRet 的值来判断 SQL 语句是否成功执行。


执行单 SQL 查询

一定要记得使用“关闭连接”命令，关闭数据库连接。这条命令的唯一属性——“数据库对象”属性，填入数据库对象 objDatabase，即可关闭数据库连接。


关闭连接

# 5. 数据处理

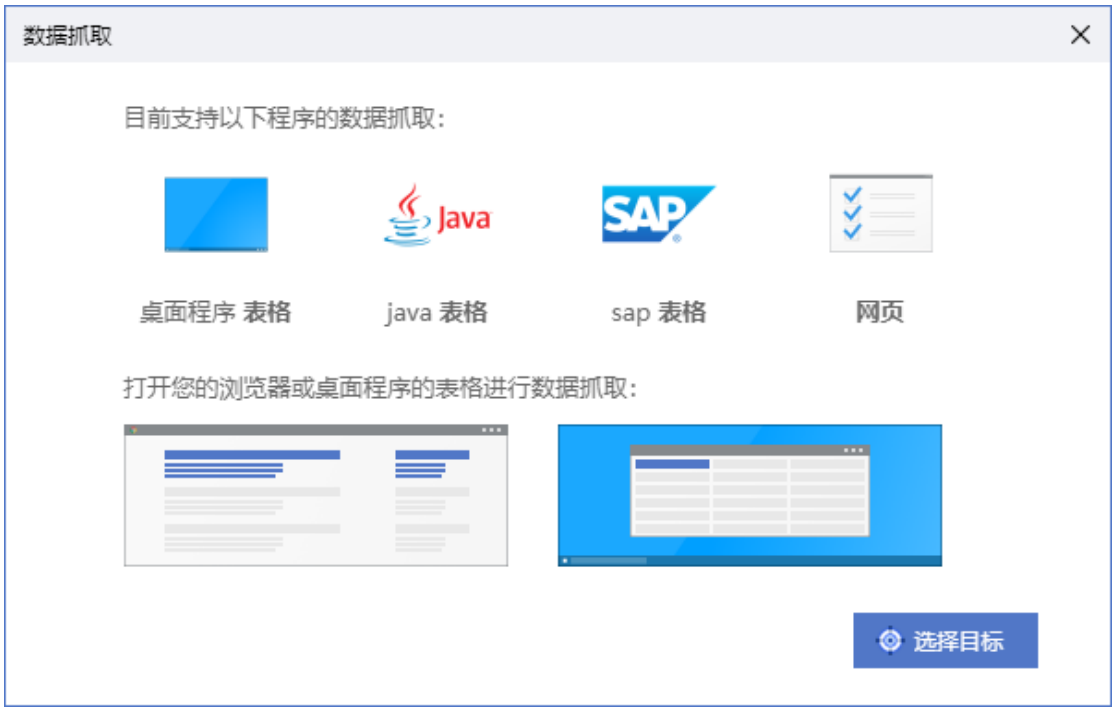
数据是信息化发展到一定阶段的必然产物。对数据进行收集、整理、加工、分析和处理，同样也是 RPA 流程中不可或缺的一环。本章以数据处理的流程顺序为主线，依次介绍数据获取、数据读取、数据处理、数据存储各个数据处理流程环节，涵盖网页数据、应用数据、文件数据等不同数据格式，JSON、字符串、正则表达式、集合、数组等多种数据处理方法。

## 5.1 数据获取方法

### 5.1.1 数据抓取

在 RPA 的流程中，经常需要从某个网页、或某个表格中获得一组数据。比如我们在浏览器中打开某个电商网站，并搜索某个商品后，希望把搜到的**每一种**商品的名称和价格都保存下来。我们固然可以用 Hibot 的“有目标命令”，逐一去网页中选择目标（包括商品名称和价格），再用获取文本的命令得到每一项的内容。但显然非常繁琐，而且在搜到的商品种类的数量不事先固定的时候，也会比较难以处理。实际上，Hibot 提供了“数据抓取”的功能，可以用一条命令，一次性的把这些内容都读出来，放在数组中。我们来看看这个功能如何使用：

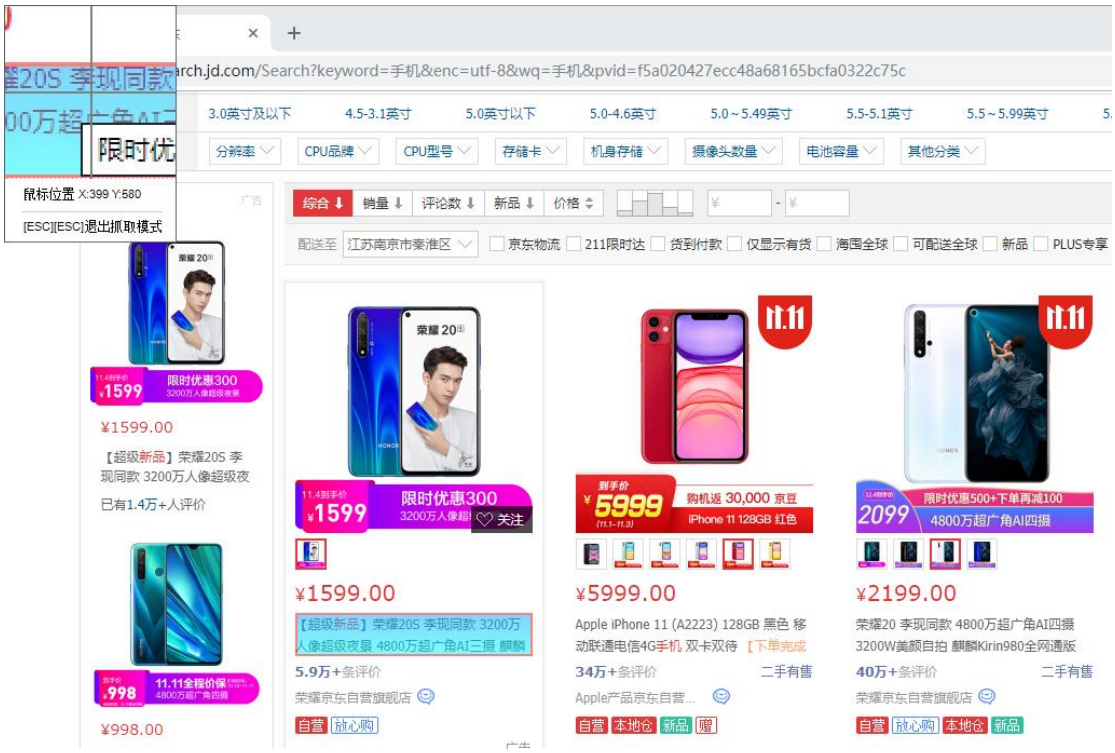
点击工具栏的“数据抓取”按钮，Hibot 将会弹出一个交互引导式的对话框，这个对话框将会引导用户完成网页数据抓取。对话框的第一步提示，Hibot 目前支持四种程序的数据抓取：桌面程序的表格、Java 表格、SAP 表格、网页，这里以网页数据抓取为例，其它三种程序的数据抓取与此类似。




开始抓取数据-选择目标

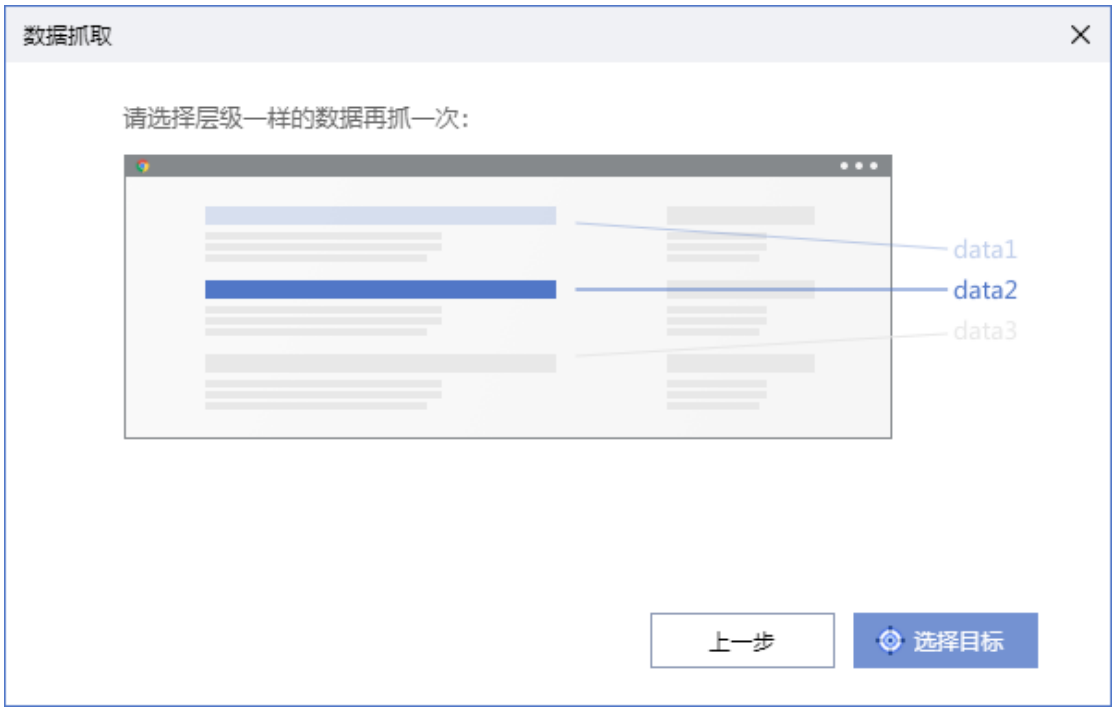
点击“选择目标”按钮，这里的“选择目标”按钮与前面我们学习的其它有目标命令中的“选择目标”按钮用法一致。需要注意的是：Hibot 并不会帮您自动打开想要抓取的网页和页面，因此在数据抓取之前，需要预先打开数据网页或桌面程序表格。这个工作可以手动完成，也可以通过 Hibot 其它命令组合完成。例如，这里演示的是抓取某网站的手机商品信息，我们可以使用“浏览器自动化”的“启动新的浏览器”命令打开浏览器并打开该网站，使用“设置元素文本”命令在搜索栏输入“手机”，使用“点击目标”命令点击“搜索”按钮。上述步骤不再展开讲解。

网页准备好后，下一步任务是在网页中定位需要抓取的数据，先抓取商品的名称，仔细选择商品名称的目标（红框蓝底遮罩框）。




选择商品名

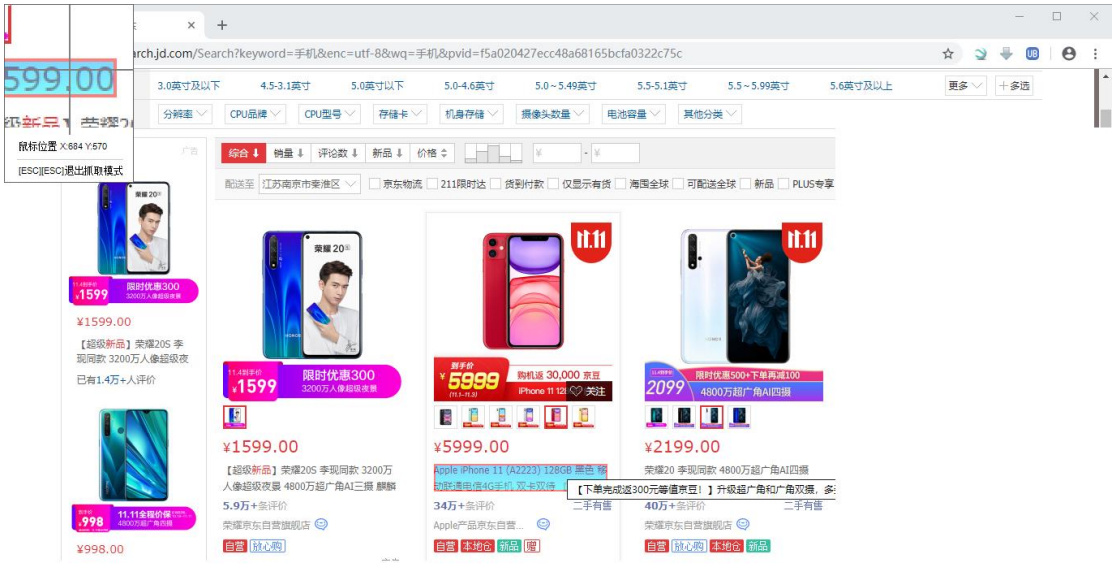
此时，Hibot 弹出提示框：“请选择层级一样的数据再抓取一次”。很多用户疑惑：什么叫层级一样的数据？为什么要再抓取一次？答案是：这是因为我们要抓取的是批量数据，必须找到这些批量数据的共同特征。第一次选取目标后，得到了一个特征，但是仍然不知道哪些特征是所有目标的共性，哪些特征是第一个目标的特性。只有再选择一个层级一样的数据并抓取一次，这样才能保留所有目标的共性，刨去每个目标各自的特性。就好比在数学中，两个点才能确定一条直线，我们只有选取两个数据，才能确定要抓取哪一列数据。




提示：选择层级一样的数据再抓取一次

定位需要抓取的数据，这里我们先抓取商品的名称，仔细选择商品名称的目标（红框蓝底遮罩框）。

再次在网页中定位需要抓取的数据，也就是商品的名称，第一次抓取的是第一个商品的名称，这次我们抓取第二个商品的名称。这里一定要仔细选择商品名称的目标，保证第二次和第一次抓取的是同一个层级的目标，因为 Web 页面的层级有时候特别多，同样一个文本标签嵌套数层目标。当然，强大而贴心的 Hibot 也会帮您做检查，这里只是先给您提个醒，可不要在 Hibot 报错的时候惊讶噢！一定是您的目标选错啦！另外，也可以选择第三个、第四个商品的名称进行抓取，这些都不影响数据的抓取结果，只要是同一层级就可以了。

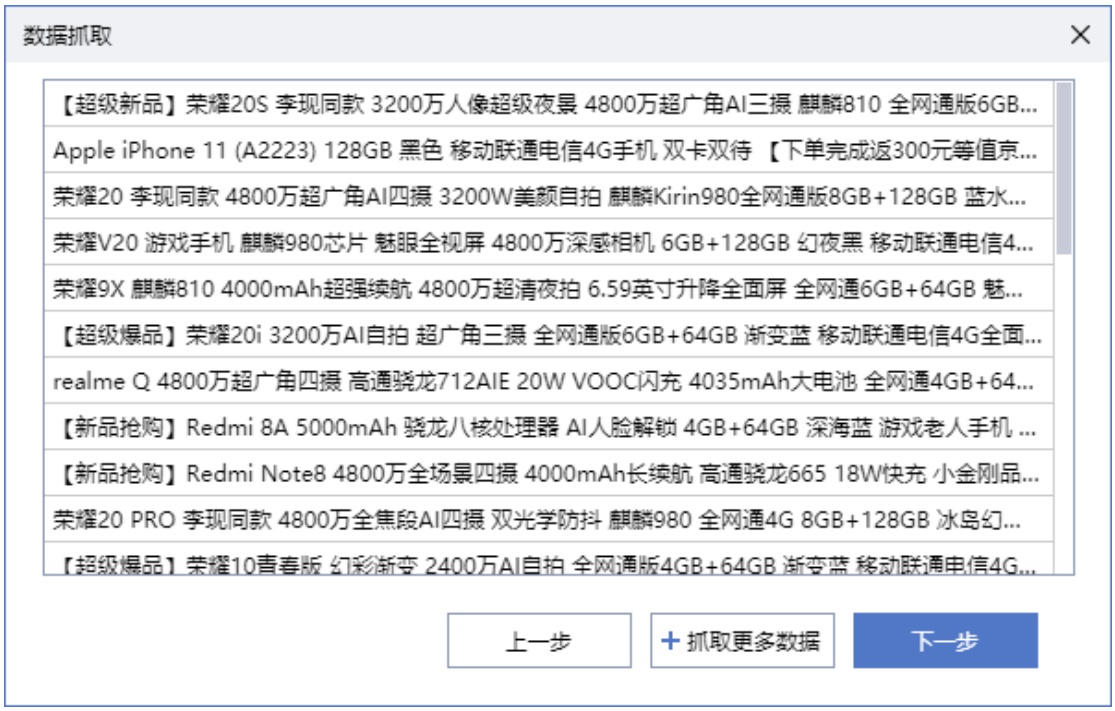



再次选择商品名

两次目标都选定完成后，Hibot 再次给出引导框，询问“只是抓取文字还是文字链接一起抓取”，这个按需选择即可。

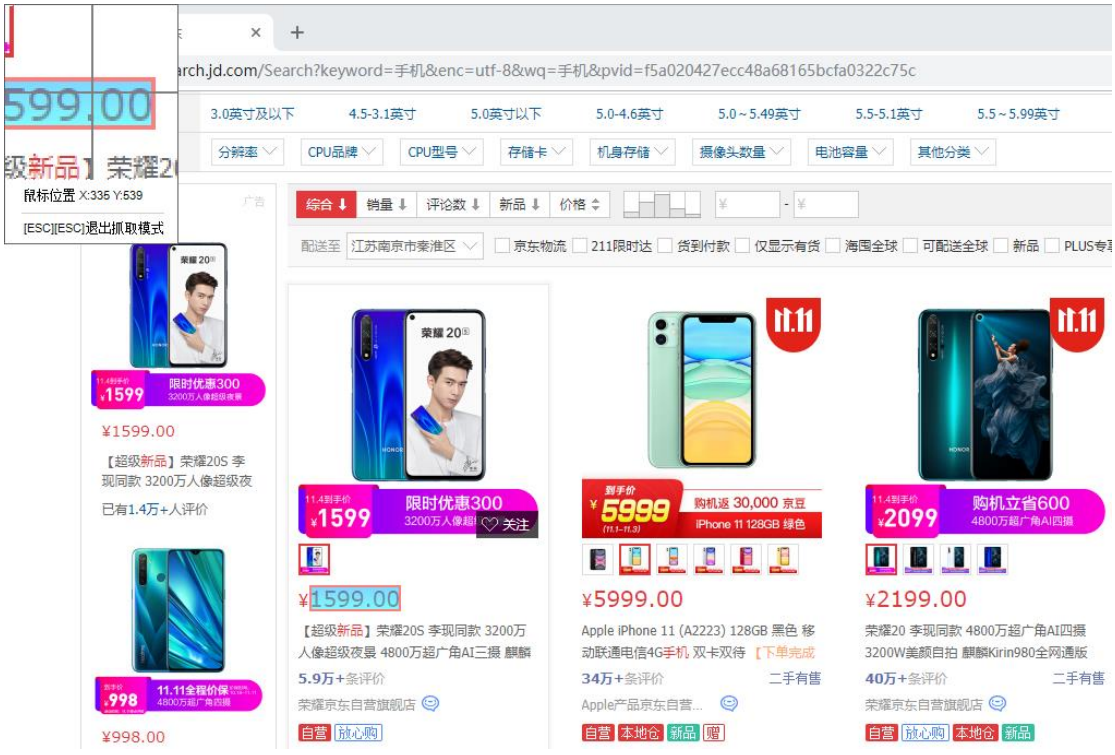

提示：您要抓取的数据是

点击“确定”按钮后，Hibot 会给出数据抓取结果的预览界面，您可以查看数据抓取结果与您的期望是否一致：如果不一致，可以点击“上一步”按钮重新开始数据抓取；如果一致，且您只想抓取“商品名称”这一个数据，那么点击“下一步”按钮即可；如果您想抓取更多的数据字段，例如，还想抓取商品的价格，那么可以点击“抓取更多数据”按钮。Hibot 会再次弹出选择目标界面。




预览抓取结果-抓取更多数据

这次我们选择的是商品价格文本标签。




选择商品价格

同样经过两次选择目标，再次预览数据抓取结果，可以看到：商品名称和商品价格已成功抓取。


再次预览抓取结果

我们可以循环使用这个方法，增加抓取的数据项，比如商品的卖家名称、评价数量等。如果不再需要抓取更多数据项了，那么点击“下一步”按钮。此时出现的引导页面询问“是否抓取翻页”，这是什么意思呢？假设把网页数据看成一个二维数据表的话，前面的步骤是增加数据表的列数，例如商品名称、价格等，而抓取翻页，是增加数据表的行数。如果只抓取第一页数据，那么点击“完成”按钮即可；如果需要抓取后面几页的数据，那么点击“抓取翻页”按钮。


抓取翻页

点击“抓取翻页”按钮，弹出“目标选择”引导框，选择 Web 页面中的翻页按钮，这里的翻页按钮为页面中的">"符号按钮。


选择翻页按钮目标

当所有步骤完成后，可以看到 Hibot 插入了一条“数据抓取”命令到命令组装区，且该命令的各个属性都已通过引导框填写完毕。例如“目标”属性的内容为：

```
{
    "html": {
        "attrMap": {
            "id": "J_goodsList",
            "tag": "DIV"
        },
        "index": 0,
        "tagName": "DIV"
    },
    "wnd": [{
        "app": "chrome",
        "cls": "Chrome_WidgetWin_1",
        "title": "*"
    }, {
        "cls": "Chrome_RenderWidgetHostHWND",
        "title": "Chrome Legacy Window"
    }]
}
```

“数据抓取”命令的某些属性还能进一步修改：“抓取页数”属性指的是抓取几页数据；“返回结果数”属性限定每一页最多返回多少结果数，-1 表示不限定数量；“翻页间隔(ms)”属性指的是每隔多少毫秒翻一次页，有时候网速较慢，需要间隔时间长一些网页才能完全打开。

### 5.1.2 通用文件处理

除了网页数据抓取，“文件”是另一种非常重要的数据源。Hibot 提供了几种格式文件的读取操作，包括通用文件、INI 格式文件、CSV 格式文件等，我们先来看一下通用文件。

在命令中心“文件处理”的“通用文件”目录下，选择并插入一条“读取文件”命令。该命令共有三个属性：“文件路径”属性，填写待读取文件的路径，这里填写的是@res"test.txt"，表示流程目录 res 子目录下的 test.txt 文件；“字符集编码”属性，选择“GBK 编码(ANSI)”（这要根据文件的编码格式来确定）；“输出到”属性，填写一个字符串变量 sRet，读取出来的文件内容将会以字符串的形式保存到这个变量中。


读取文件

通用文件只能以文本文件中的行为单位，整体地进行读取、写入和其它操作，如果需要对文件进行更加细节的操作，可以根据文件类型，选择特定的文件操作命令，例如 INI 文件、CSV 文件等。

### 5.1.3 INI 文件处理

INI 文件又叫初始化配置文件，Windows 系统程序大多采用这种文件格式，负责管理程序的各项配置信息。INI 文件格式比较固定，一般由多个小节组成，每个小节由一些配置项组成，这些配置项就是键值对。

我们来看最经典的 INI 文件操作：“读取键值”。在命令中心“文件处理”的“INI 格式”目录下，选择并插入一条“读键值”命令，这条命令可以读取指定 INI 文件中指定小节下指定键的值。该命令共有五个属性：“配置文件”属性，填写待读取 INI 文件的路径，这里填写的是@res"test.ini"，说明读取的是流程

目录下 res 子目录的 test.ini 文件，内容如下：

```
[meta]
Name = mlib
Description = Math library
Version = 1.0

[default]
Libs=defaultLibs
Cflags=defaultCflags

[user]
Libs=userLibs
Cflags=userCflags
```

“小节名”属性填写键值对的查找范围，这里填写的是“user”，说明在[user]小节查找键值对；“键名”属性填写待查找的“键”的名称，这里填写的是“Libs”，说明要查找形如“Libs=”后的内容；“默认值”属性指的是，当查找不到键时，返回的默认值；“输出到”属性填写一个字符串变量 sRet，sRet 将保存查找到的键值。


读取 INI 文件

添加一条“输出调试信息”命令，打印出 sRet，运行流程后，可以看到 sRet 的值为“userLibs”。

### 5.1.4 CSV 文件处理

CSV 文件以纯文本形式存储表格数据，文件的每一行都是一条数据记录。每条数据记录由一个或多个字段组成，用逗号进行分隔。CSV 广泛用于不同体系结

构的应用程序之间交换数据表格信息，解决不兼容数据格式的互通问题。

在 Hibot 中，可以使用“打开 CSV 文件”命令将 CSV 文件的内容读取到数据表中，然后再基于数据表进行数据处理，数据表的处理方法参见下一节。

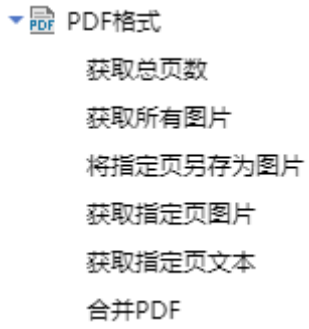
先来看“打开 CSV 文件”命令，这条命令有两个属性：“文件路径”属性填写待读取 CSV 文件的路径，这里填写的是@res"test.csv"，说明读取的是流程目录下 res 子目录的 test.csv 文件；“输出到”属性填写一个数据表对象 objDataTable，运行命令后，test.csv 文件的内容将被读取到数据表对象 objDataTable 中，我们可以添加一条“输出调试信息”命令，查看 objDataTable 对象的内容。


打开 CSV 文件

再来看“保存 CSV 文件”命令，这条命令也有两个属性：“数据表对象”属性填写上一步得到的数据表对象 objDataTable；“文件路径”属性填写保存 CSV 文件的路径，这里填写的是@res"test2.csv"，说明 objDataTable 数据表对象中的数据将被保存到流程目录下 res 子目录的 test2.csv 文件中。

## 5.1.5 PDF 文件处理

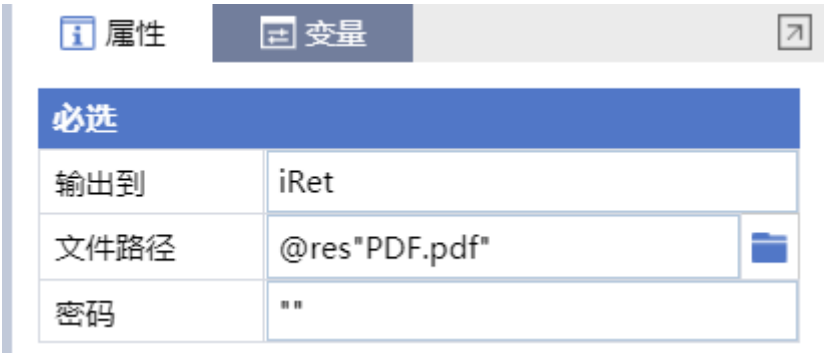
在办公场景中，PDF 格式文件是 Office 格式文件之外最常用的文件格式，因此对 PDF 文件的处理也显得非常重要。从 Hibot Designer5.0 起，Hibot 提供对 PDF 文件处理的支持。所支持的命令如下所示：




PDF 命令列表

在命令中心“文件处理”的“PDF 格式”目录下，选择并插入一条“获取总

页数”命令，这条命令可以得到指定 PDF 文件的页数。该命令共有三个属性：“文件路径”属性，填写待读取 PDF 文件的路径，这里填写的是@res"PDF.pdf"，说明读取的是流程目录下 res 子目录的 PDF.pdf 文件；“密码”属性，填写的是 PDF.pdf 文件的打开密码，如果无密码，那么保持默认值即可。运行该命令后，“输出到”属性中填写的变量名，将会保存 PDF 文件的页数。




获取 PDF 文件总页数

Hibot 还可以将 PDF 的单页转换成图片文件，选择并插入一条“将指定页另存为图片”命令，该命令共有五个属性：“文件路径”属性和“密码”属性的含义同“获取总页数”命令；“开始页码”和“结束页码”属性指定 PDF 文件的开始和结束页码，这里填写 1 和 2，表示转换第 1 页到第 2 页；“保存目录”属性填写转换后图片的保存路径，这里填写的是@res""，说明转换后图片保存到流程目录下 res 子目录。运行后，res 子目录多出两个文件：PDF_1.png 和 PDF_2.png。


将指定页另存为图片

除了处理单个 PDF 文件，Hibot 还能将多个 PDF 文件合并成一个 PDF 文件，选择并插入一条“合并 PDF”命令，该命令共有两个属性：“文件路径”属性填写需要合并的多个 PDF 文件路径。需要注意的是，该命令不支持单个 PDF 文件的合并，因此必须填写多个 PDF 文件路径，也就是需要填写一个数组，这里填写的是[@res"PDF.pdf",@res"PDF1.pdf"]，表示合并流程目录下 res 子目录的 PDF.pdf 文件和 PDF1.pdf；“保存路径”属性填写合并后的 PDF 文件路径，这里

填写的是@res"PDF2.pdf"，表示合并 PDF 保存到流程目录下 res 子目录下。


合并 PDF

## 5.2 数据处理方法

当数据完成读取后，接下来就要对数据进行处理。根据数据格式的不同，Hibot 提供了不同的数据处理方法和命令，包括通用的数据，如数据表、字符串、集合、数组、时间等，或者专有的数据，如 JSON、正则表达式等。下面分别介绍这些数据处理方法。

### 5.2.1 数据表

数据表是使用内存空间存储和处理数据的二维表格，相比存储在硬盘上的文件，内存的好处是数据处理的速度快几十上百倍，但是内存的空间相对较小。因此，一般的处理流程是：1、将需要处理的数据读取到内存中，以数据表的方式存储；2、在内存中处理数据表；3、处理完成后，将数据再次转存到硬盘上；4、再处理下一批数据。这样既可以大大加快数据处理速度，也不会受内存空间的限制。

先来看如何构建数据表。在命令中心“数据处理”的“数据表”目录下，选择并插入一条“构建数据表”命令。这条命令可以通过表头和构建数据生成数据表，该命令共有三个属性：“表格列头”属性，填写数据表的表头，这里填写的是["姓名", "科目", "分数"]；“构建数据”属性，填写数据表中的初始数据，这里填写的是[["张三", "语文", "78"],["张三", "英语", "81"],["张三", "数学", "75"],["李四", "语文", "88"],["李四", "英语", "84"],["李四", "数学", "65"]]。


构建数据表

这样，数据表就构建好了，存储到“输出到”属性中填写的变量 objDatatabl 中，如下所示：

| 姓名 | 科目 | 分数 |
| --- | --- | --- |
| 张三 | 语文 | 78 |
| 张三 | 英语 | 81 |
| 张三 | 数学 | 75 |
| 李四 | 语文 | 88 |
| 李四 | 英语 | 84 |
| 李四 | 数学 | 65 |

数据表构建完成后，可以基于数据表进行读取、排序、过滤等各种数据操作。先来看数据的排序操作。插入一条“数据表排序”命令，这条命令共有四个属性：“数据表”属性填写待排序的数据表，这里填写上一步获得的数据表对象 objDatatable；“排序列”属性表示按哪一列进行排序，这里填写的是“科目”；“升序排序”属性指的排序方法，“是”表示升序，“否”表示降序。


数据表排序

“输出到”属性填写排序之后的数据表对象，这里仍然填写 objDatatable。使用“输出调试信息”命令查看排序后的数据表，如下所示：

| 序号 | 姓名 | 科目 | 分数 |
| --- | --- | --- | --- |
| 2 | 张三 | 数学 | 75 |
| 5 | 李四 | 数学 | 65 |
| 1 | 张三 | 英语 | 81 |
| 4 | 李四 | 英语 | 84 |
| 0 | 张三 | 语文 | 78 |
| 3 | 李四 | 语文 | 88 |

再来看数据的筛选。插入一条“数据筛选”命令，这条命令共有四个属性：“数据表”属性填写待筛选的数据表，这里填写上一步获得的数据表对象objDatatable；“筛选条件”属性指的是筛选出哪些满足条件的数据，点击属性栏右边的“更多”按钮，会弹出“筛选条件”输入框。筛选条件包括为“列”、“条件”、“值”的组合，例如"科目 等于 '语文'"，表示筛选出科目为“语文”的所有数据。我们可以增加筛选条件，多个筛选条件可以是“且”的关系，也可以是“或”的关系。


数据筛选


数据筛选条件

使用“输出调试信息”命令查看筛选后的数据表，如下所示：

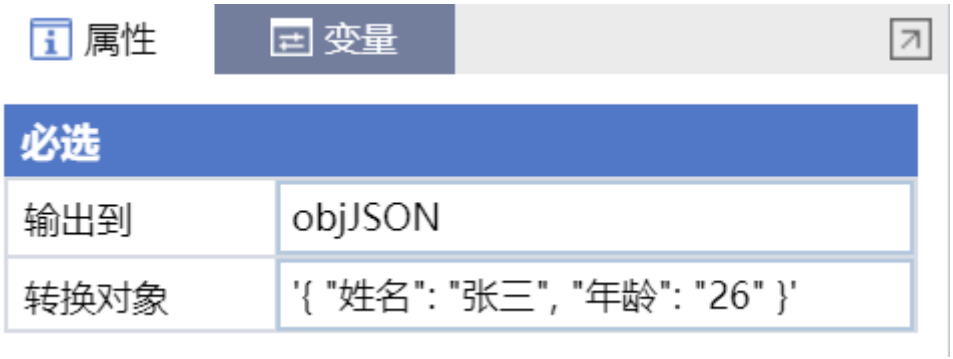
| 序号 | 姓名 | 科目 | 分数 |
| --- | --- | --- | --- |
| 0 | 张三 | 语文 | 78 |
| 3 | 李四 | 语文 | 88 |

### 5.2.2 JSON

JSON 是一种轻量级的数据交换格式，用于存储和交换文本信息。JSON 易于人阅读和编写，同时也易于机器解析和生成。JSON 在用途上类似 XML，但比 XML 更小、更快，更易解析。

Hibot 共有两条 JSON 命令，一条是“JSON 字符串转换为数据”，一条是“数据转换为 JSON 字符串”。这里的数据，其实指的是字典格式的数据。换而言之，JSON 对象等价于字典格式。JSON 字符串和 JSON 对象之间的互相转换，加上字典的一些操作，即可完成 JSON 格式的所有数据处理操作。

先来看“JSON 字符串转换为数据”命令，这条命令可以将 JSON 形式的字符串转换为 JSON 对象，该命令有两个属性：“转换对象”属性，填写 JSON 字符串，这里填写的是'{ "姓名": "张三", "年龄": "26" }'，需要特别注意的是，以往填写字符串，默认都是以双引号""作为开始和结束符号，但是这里以单引号''作为开始和结束符号。这是因为该属性中填入的是 JSON 字符串，JSON 字符串的“键名”都是以双引号""作为开始和结束符号，JSON 字符串整体以单引号''作为开始和结束符号；




JSON 字符串转换为数据

“输出到”属性，填写转换后的 JSON 对象，这里填写 objJSON。使用“输出调试信息”命令打印 JSON 对象，输出结果：{ "姓名": "张三", "年龄": "26" }，大家可能会觉得疑惑，觉得 JSON 字符串和 JSON 对象似乎也没什么区别嘛！其实区别很大，他们看起来差不多，但是一个是字符串，一个是对象，使用时差别很大，我们来看看 JSON 对象有哪些操作方式。

添加一条“输出调试信息”命令，这条命令打印 objJSON["姓名"]的值，运行后结果为"张三"，说明 JSON 对象中的数据，可以方括号的形式进行访问。

```
TracePrint(objJSON["姓名"])
```

既然能访问，应该也能修改，添加一条赋值语句，该语句将 objJSON 的"年龄"修改为 30。

```
objJSON["年龄"]="30"
```

最后，再通过“数据转换为 JSON 字符串”命令，将修改后的 JSON 对象转换为字符串。这条命令共有两个属性：“转换对象”属性，填写待转换的 JSON 对象，也就是前面一直使用的 objJSON；“输出到”属性填写一个字符串变量，该变量将会保存转换后的 JSON 字符串。使用“输出调试信息”命令查看转换后的 JSON 字符串：“{"姓名":"张三","年龄":"30"}”，可以看到，JSON 对象的内容修改成功。

### 5.2.3 字符串

字符串是系统中最常见的数据类型，字符串操作是最常见的数据操作。熟练掌握字符串操作，后续开发将会受益良多。先来看一条最经典的命令：“查找字符串”。这条命令将会查找字符串内是否存在指定的字符，该命令有五个属性：“目标字符串”属性填写被查找字符串，这里填写的是"abcdefghijklmn"；“查找内容”属性填写待查找的指定字符，这里填写的是"cd"；“开始查找位置”属性指的是从哪个位置开始查找，起始位置为 1；“区分大小写”属性指的是查找时是否区分大小写，默认为“否”；“输出到”属性填写一个变量 iRet，该变量保存查找到的字符位置。运行命令，打印变量 iRet，输出结果为 3，表明"cd"出现在"abcdefghijklmn"的第 3 位。如果要查找的字符串不存在，输出的结果将会是 0。


查找字符串

再来看一条经典的字符串操作：“分割字符串”命令。这条命令使用特定分隔符，将字符串分割为数组。比如可以用这条命令来处理前面提到的 CSV 格式文件，因为 CSV 格式文件中是有明确的分隔符的。该命令有三个属性：“目标字符串”属性填写待分割的字符串，这里填写"zhangsan|lisi|wangwu"；“分隔符”属性填写用以分割字符串的符号，这里填写的是"|"；“输出到”属性保存分割后的字符串数组到 arrRet。为了查看结果，我们再来添加一条“输出调试信息”

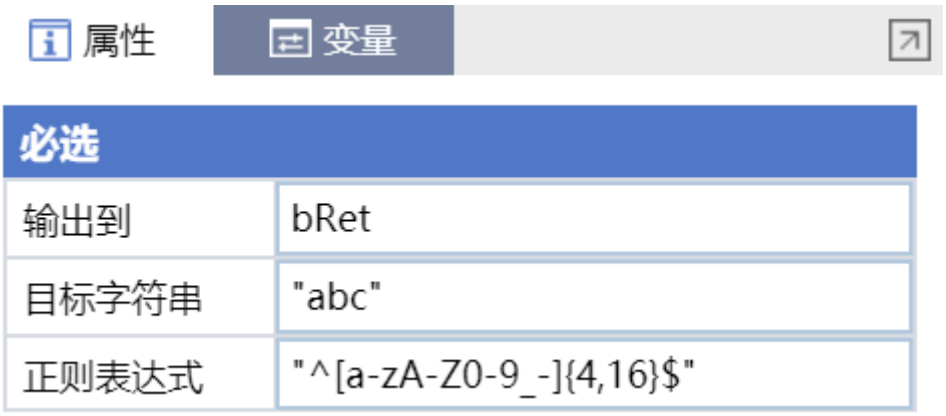
命令，输出变量 arrRet 的值，可以看到结果为[ "zhangsan", "lisi", "wangwu" ]，表明字符串"zhangsan|lisi|wangwu"通过分隔符"|"，被成功的分割为字符串数组[ "zhangsan", "lisi", "wangwu" ]。


分割字符

## 5.2.4 正则表达式

在编写字符串处理流程时，经常会需要查找和测试某个字符串是否符合某些特定的复杂规则，正则表达式就是用于描述这些复杂规则的工具，它不仅可以很方便地对单个字符串数据进行查找和测试，也可以很好地处理大量数据（如：数据采集、网络爬虫等）。

先来看“正则表达式查找测试”命令，这条命令尝试使用正则表达式查找字符串，能够找到则返回 True，找不到则返回 False，该命令可用于判断一个字符串是否满足某个条件。该命令有三个属性：“目标字符串”属性填写待测试的字符串；“正则表达式”属性填写正则表达式；“输出到”属性保存测试结果。举个例子，网站判断用户输入的注册用户名是否合法，首先将合法用户名的判断条件写成正则表达式，然后使用正则表达式去测试用户输入的字符串是否满足条件。具体来看，“正则表达式”属性填入"^[a-zA-Z0-9_-]{4,16}$"，表示注册名为 4 到 16 位，字符可以是大小写字母、数字、下划线、横线；“目标字符串”如果填入"abc_def"，测试结果返回 true，说明"abc_def"符合正则表达式。“目标字符串”如果填入"abc"或"abcde@"，测试结果返回 false，因为"abc"的长度为 3，"abcde@"中含有字符"@"，都不符合正则表达式规则。




正则表达式查找测试

再来看“正则表达式查找”命令，这条命令使用正则表达式查找字符串，找出所有满足条件的字符串。该命令有三个属性：“目标字符串”属性填写待查找的字符串；“正则表达式”属性填写正则表达式；“输出到”属性保存查找结果。举个例子，“目标字符串”属性填写网络爬虫爬回来的一段网页，如下所示：

```
'<p/>
<img src="https://avatar.csdn.net/A/4/C/3.jpg"/>
<p/>
<img src="https://g.csdnimg.cn/static/1x/11.png"/>
<p/>'
```

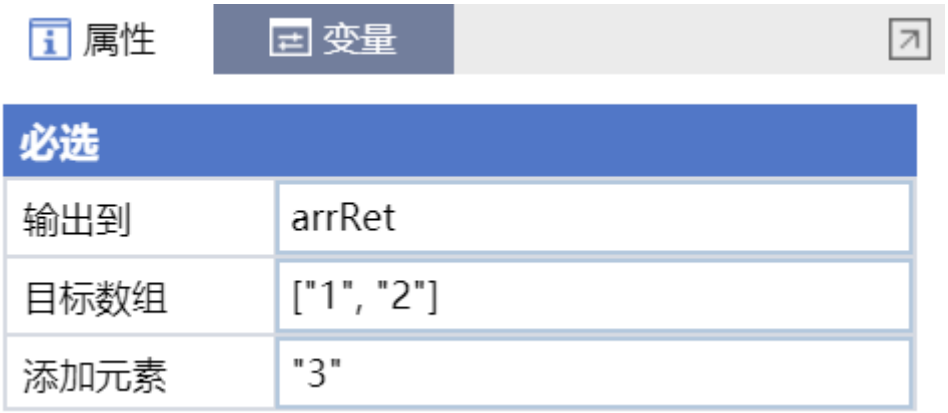
“正则表达式”属性填写"<img src=.+[png|jpg|bmp|gif]"，表示匹配以<img src=开头，以图片文件后缀 png|jpg|bmp|gif 结尾的字符串。通过这条命令，可以将爬回来网页中的所有图片链接全部抽取出来。


正则表达式查找

## 5.2.5 数组

“数组”命令主要完成数组编辑（添加元素、删除元素、截取合并数组）、

数组信息获取（长度、下标等）等功能。在命令中心“数据处理”的“数组”目录下，选择并插入一条“在数组尾部添加元素”命令，该命令可在数组的末尾添加一个元素。该命令有三个属性：“目标数组”属性，填写添加元素前的数组，这里填写["1", "2"]；“添加元素”属性填写待添加的元素，这里填写"3"；“输出到”属性保存添加后的数组变量，打印该变量，预期输出结果为["1", "2", "3"]。




在数组尾部添加元素

再来看“过滤数组数据”命令，这条命令可以快速对数组中的元素进行筛选，留下或者剔除满足条件的元素。该命令有四个属性：“目标数组”属性，填写待过滤的数组，这里填写的是["12", "23", "34"]；“过滤内容”属性填写按照什么条件过滤数组，这里填写"2"，表示数组元素只要是字符串，并且包含了"2"就满足条件；“保留过滤文字”属性有两个选项，“是”表示满足条件的数组元素将会保留，剔除不满足条件的元素；“否”表示满足条件的数组元素将会剔除，保留不满足条件的元素。“输出到”属性保存处理后的数组 arrRet。

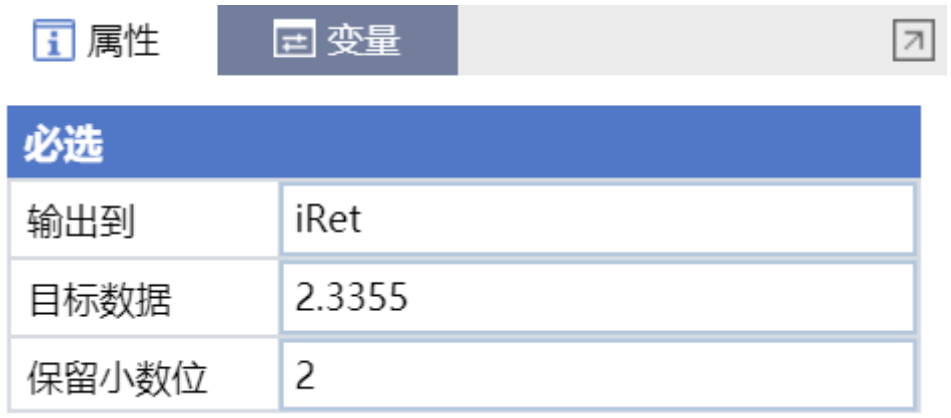
我们尝试一下，对“保留过滤文字”的属性选择“是”，并输出过滤后的数组变量 arrRet，输出结果为[ "12", "23" ]，包含"2"字符串的数组元素都被保留；“保留过滤文字”属性选择“否”，打印过滤后的数组变量 arrRet，输出结果为[ "34" ]，包含"2"字符串的数组元素都被剔除。


过滤数组数据

### 5.2.6 数学

数学操作命令位于命令中心“数据处理”的“数学”目录下，包括各种数学操作，这些命令相对独立，因此这里只选择一个讲解，其它命令的使用方式类似，在此不再赘述。

选择并插入一条“取四舍五入值”命令，这条命令可对数字取四舍五入。该命令的属性共有三个：“目标数据”属性填写需要四舍五入的数字；“保留小数位”属性填写小数保留位数；“输出到”属性保存四舍五入后的结果。



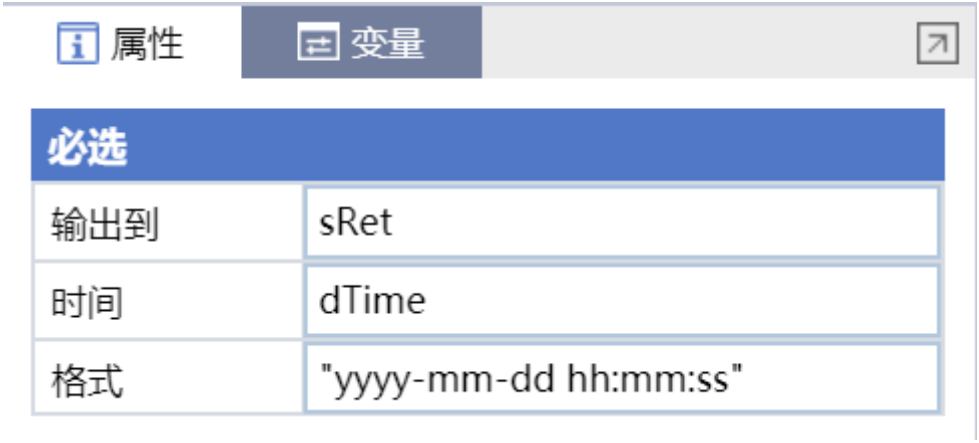

取四舍五入值

### 5.2.7 时间

时间操作命令主要包括时间数据和字符串的互相转换，以及对时间数据的各种操作。首先来看如何获取当前时间，在命令中心“数据处理”的“时间”目录下，选择并插入一条“获取时间”命令。这条命令可以获取从 1900 年 1 月 1 日起到现在经过的天数，该命令只有一个“输出到”属性，保存当前时间，这里填

写 dTime。运行流程后，“输出调试信息”命令打印调试信息：43771.843969907，说明从 1900 年 1 月 1 日到现在，已经过去了 43771.843969907 天，大家可以大致估算一下是否正确。

得到时间数据后，可以通过“格式化时间”命令将时间数据转换成各种格式的字符串。“格式化时间”命令有三个属性：“时间”属性填写刚刚得到的时间数据 dTime；“格式”属性填写时间格式，其中年(yyyyy)占 4 位、月(mm)、日(dd)、24 小时(hh)、分(mm)、秒(ss)都占 2 位，例如"yyyy-mm-dd hh:mm:ss"填写时间后转换为："2019-11-02 20:29:58"；“输出到”属性保存格式化时间的结果。




格式化时间

除了将时间数据转换成各种格式的字符串，也可以直接获取时间数据的某一分项。例如可以使用“获取月份”命令获取时间数据 dTime 中的月份，以此类推，其它命令类似。

### 5.2.8 集合

集合操作命令主要包括集合的创建、集合元素的增删、集合间的操作等。首先来看创建集合。在命令中心“数据处理”的“集合”目录下，选择并插入一条“创建集合”命令。该命令只有一个“输出到”属性，将创建集合的结果赋值给 ObjSet 对象。

接着，我们往 ObjSet 这个集合中写入元素，插入一条“添加元素到集合”命令。该命令有两个属性：“集合”属性填写上一步创建的集合对象 ObjSet；“添加元素”属性填写集合元素，可以是数字、字符串等常量，也可以是变量。

同一个集合中，能否既有数字元素，又有字符串元素呢？答案是肯定的！我们可以调用两次“添加元素到集合”命令，一次插入 1，一次插入“2”。运行后打印调试信息，可以看到两个元素都成功插入集合。

最后来看多个集合之间的操作，以取集合的并集为例。通过插入元素构建出两个集合，一个为{1, "2"}，另一个为{"1", "2"}。添加一条“取并集”命令。

该命令有三个属性：“集合”属性和“比对集合”分别填写需要合并的两个集合；“输出到”属性填写合并之后的集合变量。运行后打印调试信息，可以看到合并之后集合变为{1, "1", "2"}，这说明并集剔除了重复元素"2"，1 和“1”一个是数值，一个是字符串，不属于重复元素，因此同时选入并集。

以上内容的关键源代码如下：

```
ObjSet=Set.Create()
Set.Add(ObjSet,1)
Set.Add(ObjSet,"2")
TracePrint(objSet)

ObjSet2=Set.Create()
Set.Add(ObjSet2,"1")
Set.Add(ObjSet2,"2")

objSetRet = Set.Union(ObjSet,ObjSet2)
TracePrint(objSetRet)
```

# 6. 人工智能功能

## 6.1 NLP（自然语言处理）

### 6.1.1 NLP 的用途

自然语言处理（NLP）就是在计算机语言和人类语言之间沟通的桥梁，以实现人机交流的目的。既然不同人类语言之间可以有翻译，那么人类和机器之间是否可以通过“翻译”的方式来直接交流呢？当然可以！NLP 就是人类和机器之间沟通的桥梁！

NLP 是人类和机器之间沟通的桥梁

在人工智能出现之前，计算机已经能够智能处理结构化的数据（例如 Excel 里的数据、数据库中的数据）。但是网络中大部分的数据都是非结构化的，例如：网页、图片、音频、视频等等。在非结构数据中，文本的数量是最多的，它虽然没有图片和视频占用的空间大，但它的信息量是最大的。想要计算机处理这些非结构化的文本数据，理解这些文本信息，就需要用到 NLP 技术。


结构化与非结构化数据

## 6.1.2 NLP 的核心任务

自然语言就是大家平时在生活中常用的表达方式，大家平时说的「讲人话」就是这个意思。

自然语言：我背有点驼　非自然语言：我的背部呈弯曲状

NLP 有 2 个核心的任务：

- 自然语言理解（NLU）
- 自然语言生成（NLG）

## 6.1.3 自然语言理解

自然语言理解就是希望机器像人一样，具备正常人的语言理解能力，由于自然语言在理解上有很多难点，所以自然语言理解方面，至今机器还远不如人类的表现。

自然语言理解的 5 个难点：

**难点 1：语言的多样性**

自然语言没有什么通用的规律，你总能找到很多例外的情况。

另外，自然语言的组合方式非常灵活，字、词、短语、句子、段落…不同的组合可以表达出很多的含义。例如：

```
我要听大王叫我来巡山
给我播大王叫我来巡山
我想听歌大王叫我来巡山
放首大王叫我来巡山
给唱一首大王叫我来巡山
放音乐大王叫我来巡山
放首歌大王叫我来巡山
给大爷来首大王叫我来巡山
```

**难点 2：语言的歧义性**

如果不联系上下文，缺少环境的约束，语言有很大的歧义性。例如：

```
我要去拉萨
```

- 需要火车票？
- 需要飞机票？
- 想听音乐？
- 还是想查找景点？

**难点 3：语言的鲁棒性**

自然语言在输入的过程中，尤其是通过语音识别获得的文本，会存在多字、少字、错字、噪音等问题。例如：

```
大王叫我来新山
大王叫让我来巡山
大王叫我巡山
```

**难点 4：语言的知识依赖**

语言是对世界的符号化描述，语言天然连接着世界知识，例如：

```
大鸭梨
```

除了表示水果，还可以表示餐厅名

7 天

可以表示时间，也可以表示酒店名

晚安

有一首歌也叫《晚安》

**难点 5：语言的上下文**

上下文的概念包括很多种：对话的上下文、设备的上下文、应用的上下文、用户画像…

用户：买张火车票
回答：请问你要去哪里？
用户：宁夏
用户：来首歌听
回答：请问你想听什么歌？
用户：宁夏

下面用一个具体的案例来深度说明一下自然语言理解：

在生活中，如果想要订机票，人们会有很多种自然的表达：

“订机票”；

“有去上海的航班么？”；

“看看航班，下周二出发去纽约的”；

“要出差，帮我查下机票”；

等等等等……

可以说“自然的表达” 有无穷多的组合，都是在代表 “订机票” 这个意图的。而听到这些表达的人，可以准确理解这些表达指的是“订机票”这件事。

而要理解这么多种不同的表达，对机器是个挑战。在过去，机器只能处理“结构化的数据”（比如关键词），也就是说如果要听懂人在讲什么，必须要用户输入精确的指令。

所以，无论你说“我要出差”还是“帮我看看去北京的航班”，只要这些字

里面没有包含提前设定好的关键词“订机票”，系统都无法处理。而且，只要出现了关键词，比如“我要退订机票”里也有这三个字，也会被处理成用户想要订机票。

自然语言理解这个技能出现后，可以让机器从各种自然语言的表达中，区分出来，哪些话归属于这个意图；而那些表达不是归于这一类的，而不再依赖那么死板的关键词。比如经过训练后，机器能够识别“帮我推荐一家附近的餐厅”，就不属于“订机票”这个意图的表达。

并且，通过训练，机器还能够在句子当中自动提取出来“上海”，这两个字指的是目的地这个概念（即实体）；“下周二”指的是出发时间。

这样一来，“机器就能听懂人话啦!”。

## 6.2 RPA 与 NLP

RPA（机器人流程自动化）和 AI（人工智能）是近期两个热门的领域和概念，很多人容易把两者混为一谈，其实，RPA 与 AI 的关系还是比较清晰和简单。

首先，二者的区别还是较为明显，有着不同的技术特征。一个是与“思考”和“学习”有关，一个则是与“做”有关。

AI 结合机器学习和深度学习，具有很强的自主学习能力。通过计算机视觉、语音识别、自然语言处理等技术拥有认知能力，可以通过大数据不断矫正自己的行为，从而有预测、规划、调度以及流程场景重塑的能力。

RPA 作为软件机器人，其需要依靠固定的脚本执行命令，模拟用户手工操作及交互，进行基于明确规则、重复、机械性的劳作，并以外挂形式部署在客户原有的系统上。RPA 和那些自动化生产线上的工业机器人较为相似，只能死板地按照人类给它规定的程序工作。

其次，RPA 和 AI 都有替代人工劳动的功能，AI 与 RPA 的关系，就好比人类大脑和手脚的关系。

在具体应用上二者各司其职，密不可分。

- RPA 倾向于重复地执行命令，AI 更倾向于发出命令。
- RPA 机器人能够将简单的工作自动化，并为 AI 提供大数据。
- AI 能够根据 RPA 提供的数据进行模仿并改进流程。
- RPA 以流程为中心，AI 以数据为中心。

有了 AI 的赋能，PRA 就可以提升自身的智能化水平，因此智能自动化（IA）、智能流程自动化（IPA）也频繁出现在一些 RPA 厂商的宣传用语中。我们可以以 AI 的赋能程度，将 RPA 简单地划分为三个阶段：

- RPA 阶段，没有 AI 的参与，着重应付结构化、高度重复性的工作。主要涉及批处理、桌面自动化等技术。
- 智能自动化（IA）阶段，可以处理非结构化、一定规律性的任务。主要涉及自然语言处理、深度学习等技术。此阶段也可被看成，认知性 RPA 阶段，即“RPA 4.0”。
- 人工智能（AI）阶段，非结构化自由形式。主要涉及深度学习机器学习，认知计算等技术。

而在 AI 技术中，NLP 和 OCR 是最实用的两项技术。

NLP 可以实现常见文本的理解和抽取，实现文本（财务报表、合同、招股说明书、报告等）的智能抽取。例如，金融行业和政府机构都有大量的资料报送和审查场景，存在大量非结构化的数据。一张单据上，无论填写者用“供应商”还是“甲方”，表达的意思是一样的，对于 RPA 机器人，需要深入理解“供应商”还是“甲方”的含义。

## 6.3 Hibot 中的 NLP

Hibot 目前版本中，提供了两个最基本的自然语言处理命令，一个是 “分词&词性标注”，一个是 “实体抽取”。例如我们处理如下句子：

```
帮张三订一张 9 月 10 日去上海的机票″
```

“分词&词性标注”命令将上述句子解析如下，可以看到，将这句话的每个词都解析正确。

```
[{
    ″idx_start″: 0,
    ″pos″: ″vl″,
    ″text″: ″帮″
}, {
    ″idx_start″: 1,
    ″pos″: ″nh″,
    ″text″: ″张三″
}, {
    ″idx_start″: 3,
    ″pos″: ″vl″,
    ″text″: ″订″
}, {
    ″idx_start″: 4,
    ″pos″: ″m″,
    ″text″: ″一″
}, {
```

```
    "idx_start": 5,
    "pos": "q",
    "text": "张"
}, {
    "idx_start": 6,
    "pos": "m",
    "text": "9 月"
}, {
    "idx_start": 8,
    "pos": "nt",
    "text": "10 日"
}, {
    "idx_start": 11,
    "pos": "vl",
    "text": "去"
}, {
    "idx_start": 12,
    "pos": "ns",
    "text": "上海"
}, {
    "idx_start": 14,
    "pos": "u",
    "text": "的"
}, {
    "idx_start": 15,
    "pos": "n",
    "text": "机票"
}]
```

“实体抽取”命令可以将一个句子中的实体都抽取出来，假设某句话为：“下周一北京到上海的机票是一千两百块钱”，则实体抽取源代码如下：

```
dim arrEntity = NLP.Extract("下周一北京到上海的机票是一千两百块钱")
TracePrint arrentity
```

运行后结果如下：

```
[{
"idx_start" : 0,
"name" : "sys.date",
"standard_value" : "2019-12-30",
"text" : "下周一"
},
```

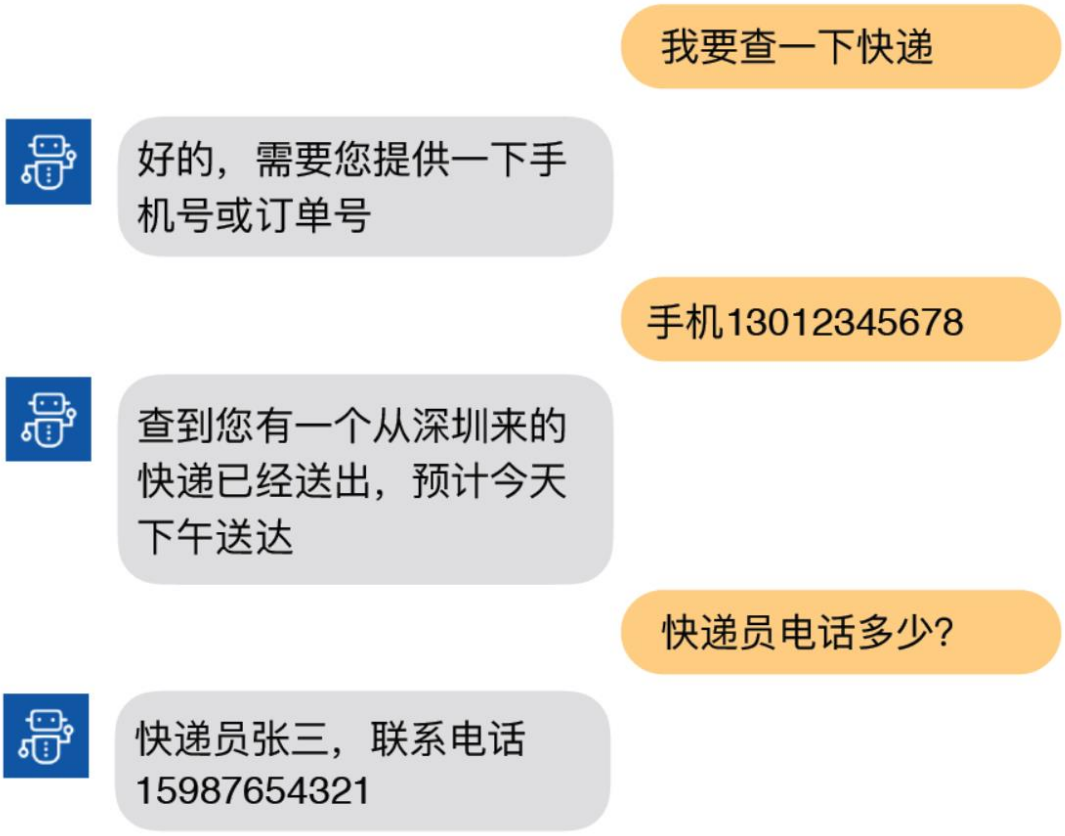
```
{
"idx_start" : 3,
"name" : "sys.poi",
"standard_value" : "北京",
"text" : "北京"
},
{
"idx_start" : 3,
"name" : "sys.city",
"standard_value" : "北京",
"text" : "北京"
},
{
"idx_start" : 6,
"name" : "sys.poi",
"standard_value" : "上海",
"text" : "上海"
},
{
"idx_start" : 6,
"name" : "sys.city",
"standard_value" : "上海",
"text" : "上海"
},
{
"idx_start" : 12,
"name" : "sys.price",
"standard_value" : "1200.00",
"text" : "一千两百块钱"
}]
```


可以看到，上述句子中的 “下周一”、“上海”、“北京”、“一千两百块钱”等实体都被正确抽取。Hibot 中已内置支持 50 多种开箱即用的系统实体，包括：日期、时间、节日、姓名、手机号、身份证、邮箱、国家、省份、城市、地址、价格、重量、长度等。

虽然这些功能还比较简单，而且还必须通过互联网向服务器发送请求，才能完成一次分词或实体抽取操作。但这却是复杂 NLP 功能的基础。如果您的业务流程中还需要更复杂的 NLP 技术的支持，例如理解语言中的意图等，可以和 Hibot 官方联系，进行定制化的开发和训练。定制化的版本也可以选择部署在您公司内部的服务器上，即使不连接互联网也能完成任务。

比如，下图中的这段人机对话，就是 Hibot 官方团队中的几位 NLP 专家的研发成果。机器人不仅能准确理解用户的意图，还能记住用户前面所述的意图哦！

这样聪明的机器人和 RPA 结合起来，就能完成更多、更复杂、也更有趣的任务。




人机对话的机器人

## 6.4 OCR

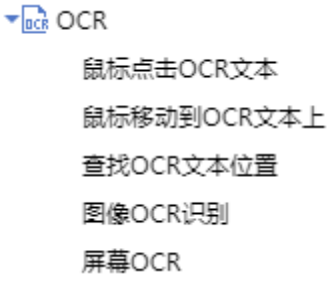
我们在前面的内容中提到，有些情况是无法获取界面元素的。此时，使用“图像”类命令，可以找到准确的操作位置。但还不能像有目标的命令那样，把界面元素中的内容读出来。

比如，著名的游戏平台 Steam，其界面使用了 DirectUI 技术绘制，我们无法获得其中的任何文字（虽然这些内容用肉眼很容易看到），如图所示。此时，就需要祭出 Hibot 的“OCR”类命令了。


很难直接获取 Steam 界面中的文字

OCR 的全称是“光学字符识别”，这是一项历史悠久的技术，早在上个世纪，OCR 就可以从纸质的书本中扫描并获得其中的文字内容。如今，OCR 的技术也在不断演进，已经融入了流行的深度学习等技术，识别率不断提高。我们现在用 OCR 去识别屏幕上的文字，由于这些文字不像纸质书本一样存在印刷模糊、光线不好等问题，所以识别率是非常高的。

Hibot 中包含了以下的 OCR 命令：




Hibot 的 OCR 命令

其中，“鼠标点击 OCR 文本”、“鼠标移动到 OCR 文本上”、“查找 OCR 文本位置”三条命令类似于“图像”类中的“点击图像”、“鼠标移动到图像上”，“查找图像”命令，只不过不需要传入图像了，只需要在属性中标明要找的文字即可。

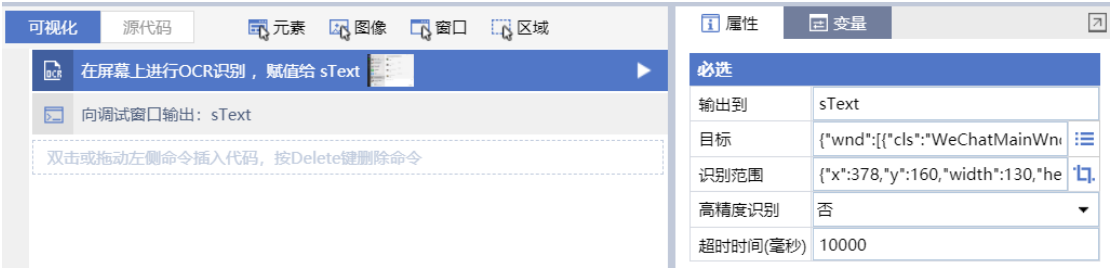
“图像 OCR 识别”命令和“屏幕 OCR”命令类似，只不过前者需要提供一个图像文件，后者需要提供一个窗口、以及窗口中的一个区域，Hibot 会在流程运行到这一行的时候，自动在窗口的指定区域截图并保存为一个文件，然后采用和前者一样的方式去执行。

我们先试一下“屏幕 OCR”命令。双击或拖动插入一条“屏幕 OCR”命令，点击命令上的“查找目标”按钮（此时 Hibot Designer 的窗口会暂时隐藏）；把鼠标移动到 Steam 的登录窗口上，此窗口会被红框蓝底的遮罩遮住；此时拖动鼠标，划出一个要进行文字识别的区域，这个区域会用紫色框表示。如下图所示。


选择 OCR 目标

这样的一条命令，会在运行的时候，自动找到 Steam 的登录窗口，并在紫色框指定的位置（相对于窗口的位置）截图，然后识别截图里面的文字，最后把识别到的文字保存在变量 sText 中。

OCR 命令完成之后，为了看到效果，最好加入一条“输出到调试窗口”命令，并指定输出变量 sText。注意 sText 是变量名，而不是字符串，所以两边不加双引号。




完成一条 OCR 命令

运行这个流程块，即可看到效果。只要 Steam 的登录窗口存在，且窗口大小没有发生变化，就能识别出我们所划的区域中的文字“账户名称”。