

UiBot开发者指南（初级）

生成时间：2021-12-06 15:12:46

目录

1 RPA简介	3
1.1 RPA基础知识	3
1.2 RPA平台介绍	4
1.3 UiBot发展历程	5
1.4 RPA和AI	6
1.5 后续内容	7
2 基本概念	9
2.1 概念	9
2.2 流程图	9
2.3 可视化视图	11
2.4 源代码视图	13
2.5 小结	15
2.6 进阶内容	16
3 界面元素自动化	25
3.1 界面元素	25
3.2 目标选取	27
3.3 目标编辑	29
3.4 界面元素操作	32
3.5 UI分析器	38
3.6 安装扩展	40
4 界面图像自动化	47
4.1 为什么不能使用界面元素?	47
4.2 无目标命令	48
4.3 图像命令	49
4.4 实用技巧	53
4.5 智能识别	55

5	软件自动化	59
5.1	Excel自动化	59
5.2	Word自动化	66
5.3	邮件客户端自动化	70
5.4	浏览器自动化	73
5.5	数据库自动化	74
6	逻辑控制	78
6.1	条件分支	78
6.2	循环结构	80
6.3	循环的跳出	82
7	流程和任务管理	85
7.1	RPA的两种工作模式	85
7.2	流程管理	86
7.3	人机交互模式	89
7.4	无人值守模式	93
7.5	机器人的协同	97
8	结束语	100
	附录：编程基础知识	101
8.1	数据	101
8.2	数据类型	101
8.3	变量	102
8.4	表达式	102
8.5	条件判断	103
8.6	循环	103

1 RPA简介

欢迎您选择使用UiBot!

如果您听说过RPA，或者做过RPA方面的项目，那么恭喜您做出了一个明智的选择！因为截至发稿时间为止，在诸多国内RPA平台中，UiBot的设计理念和技术实现都处于遥遥领先的地位，即使与国外产品相比也不遑多让。如果您对RPA已经比较熟悉，您可以跳过本章，直接开始感受UiBot的魅力。

如果您没有听说过RPA，俗话说：磨刀不误砍柴功，那就先通过本章，了解一下RPA吧！

1.1 RPA基础知识

我们平时在使用计算机软件时，常常遇到大量机械重复而又繁琐的操作。如果都依靠人工来操作，不仅很容易感到疲劳和厌烦，还经常会出错。比如，对于财务工作人员来说，常常需要使用网上银行，给很多客户转账，转一笔账也许并不算麻烦，但如果每天要转成百上千笔账，那就是一件摧残人性的工作了。更可怕的是，有时候头晕眼花，在“金额”一栏中多输入一个零……不必吃惊，这样的事情已经发生过很多次了。

为了应对财务领域这种简单重复的体力劳动，在2017年，作为财务方面的绝对权威，国际著名的四大会计师事务所（安永、普华永道、德勤、毕马威）先后把国外已有初步应用的“财务机器人”概念引入中国，试图制造一个“软件机器人”，或者叫“数字化劳动力”，自动完成这些计算机上的机械重复工作。

后来，大家逐渐发现：其实“软件机器人”在财务之外的领域也大有可为，物流、销售、人力资源等很多工作领域中都大量存在简单重复的体力劳动，而偷懒是人类共同的天性，于是，这些软件机器人很快在国外和国内的诸多领域都得到了大量应用，而且有愈演愈烈之势。当然，此时已不局限于财务领域了，“财务机器人”的说法显得有些不合时宜。于是，从美国开始，大家普遍把制造各种“软件机器人”当成了一个新的行业，称之为Robotic Process Automation，中文翻译为机器人流程自动化，简称**RPA**。

实际上，软件的流程自动化并不是一个新概念，比如微软的Office办公软件中，早在二十多年前就自带了“宏”（Macro）的功能，可以自动化操作Office来工作。

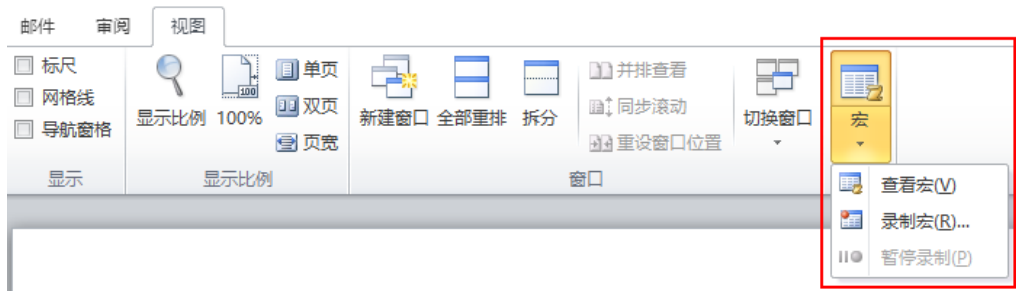


图 1: Office的“宏”功能

但是，RPA和早期的软件流程自动化有所不同，RPA强调对已有系统的“无侵入”，这是什么意思呢？就是说，如果一个软件本身不支持自动化的功能，那么RPA就可以大显身手了。RPA不需要对这个软件进行任何修改，而是通过模拟人的阅读和操作软件的方式，让这个软件实现自动化。不仅如此，如果有两

个、三个甚至更多的软件需要一起工作，如果采用传统的Office“宏”的方式，那就必须让这些软件都进行修改，实现一套统一的接口才行。有一个不改，流程就不能自动化，这显然有点儿强人所难，毕竟这些软件很可能不是同一个厂商开发的！但是，如果采用RPA，这些软件 **全部** 都不需要经过任何修改，RPA会制造一个“软件机器人”，模拟人的阅读和操作，让“软件机器人”自动完成这个流程，这种“无侵入”的特点是RPA的核心魅力之一。

RPA即将成为一个新的行业，这个说法一点儿也不夸张。在国外，RPA领域已经出现了市值数百亿美金的上市企业，专门为其它行业提供RPA的产品和服务。国外的产品做得很不错，能够较好的满足RPA的需求，但是到了中国，或多或少还是有些“水土不服”，比如对中文的支持，对国内银行网银等软件的支持，都差强人意；并且这些公司还缺乏本土化的技术服务，出了问题不能及时得到响应。在这种情况下，选择北京来也网络科技有限公司出品的UiBot来做RPA，就是您最明智的选择了。

1.2 RPA平台介绍

UiBot是一种RPA平台，那么什么是RPA平台呢？且听我慢慢解释：

为了实现RPA，即机器人操作的流程自动化，打造一个前面所说的“软件机器人”，通常需要如下几个步骤：

1. 梳理和分析现有的工作流程，看看什么地方可以用“软件机器人”来改造，实现自动化；
2. 从技术上实现“软件机器人”，让它能够阅读和操作流程中所涉及到的所有软件；
3. 把“软件机器人”部署到实际工作环境中，启动机器人开始工作，监控机器人的运行状况，如果出现问题还要及时处理。

第一步通常由业务专家来做，比如在财务领域，就需要财务专家来进行财务工作流程的梳理和分析；第二步通常由IT专家来做，对于这些编程高手来说，用类似Python这样强大的编程语言来实现一个模拟人类工作的机器人，并非难事；第三步通常由普通工作人员来做，只要按一个按钮，启动机器人，就可以在旁边喝茶刷手机了，一切都很美好，对不对？

可是事实并非如此。第一步，业务专家梳理和分析流程，没问题。第二步，问题来了，术业有专攻，IT专家常常沉浸在数字化的世界里，对业务一窍不通，根本不理解业务专家梳理的流程是怎么回事儿，无从下手！第三步，问题更大了，普通工作人员又不懂IT，让他们去启动机器人还行，出现问题怎么解决？只能呼叫IT专家紧急支援，如果支援不及时，可能就耽误了工作。

比如，笔者自己是IT技术出身，见了财务领域的“台账”、“交易性金融资产”这样的名词就头大；反之，笔者耳熟能详的“句柄”、“线程”等概念，对于大多数财务专家来说，恐怕也是一头雾水，更别提普通工作人员了。

怎么办呢？RPA的理念是：

- 打造RPA平台，把一些常见的RPA功能做成半成品，就像方便面等方便食品一样；
- 让业务专家站在RPA平台这个巨人的肩膀上，自己就能做出机器人，难度就像泡一碗方便面一样；
- 让普通工作人员也能看懂机器人的大致原理，必要的时候还可以修改，难度就像给方便面加一点点调料一样，根本不需求助IT专家；
- 从此，“软件机器人”的生产过程不再需要IT专家参与，世界重归美好！

为了实现上述理念，一般的RPA平台至少会包含以下三个组成部分：

- 开发工具：主要用来制作“软件机器人”，当然也可以运行和调试这些机器人；
- 运行工具：当开发完成后，普通用户使用RPA平台，来运行搭建好的机器人，也可以查阅运行结果；
- 控制中心：当需要在多台电脑上运行“软件机器人”的时候，可以对这些“软件机器人”进行集中控制，比如统一分发，统一设定启动条件等。

啰嗦了这么多，终于带出“**RPA平台**”的概念了。所谓RPA平台，就是把“软件机器人”分解成很多零件，让不懂IT的业务专家能以搭积木的方式，把这些零件在自己的工作台上搭起来，而不需要IT人员的参与，让普通工作人员能看到机器人的基本原理和执行的情况，还能进行简单的维护。

所以，RPA平台的关键指标是：

- 要足够**强大**，零件数量要多，复杂的场景也能应对；
- 要足够**简单**，不需要IT专家的参与，普通人就可以轻松掌握；
- 要足够**快捷**，普通人稍微熟练一些以后，可以用最便捷的方式，快速实现自己的机器人。

为了实现这些指标，各种RPA平台作出了很多努力，但效果仍然差强人意。主要是因为这些指标往往是相互矛盾的，按下葫芦浮起瓢，想要强大就很难简单，想要简单又很难快捷。比如有的RPA平台直接让大家用Python编程语言来实现RPA，因为Python本身就足够强大，可是术业有专攻，业务专家和普通用户要精通Python，恐怕不那么容易。所以，这样“剑走偏锋”的RPA平台输掉了简单和快捷这两项指标，结果自然是“走火入魔”。

UiBot也是一种RPA平台，为了在RPA平台的这三个关键指标上取得平衡，UiBot作出了大量的努力。有些努力您能够从软件界面中看到，有些努力您可能看不到，比如针对一些关键的设计理念，UiBot的设计人员曾花费半年的时间深入调研和反复讨论，几易其稿，才终于拿出一个相对完善的方案。所以，我们很自信地说UiBot在国内的RPA平台中处于遥遥领先的地位，是因为产品经过精心打磨，三个关键指标都达到了比较满意的程度。

当然，仅凭努力还不够。实际上，UiBot的核心团队从2001年开始，就在做流程自动化方面的事情了，到今天为止已经过去了二十余年，所以才能积累丰富的经验，在一些关键点的设计和研发上把握得游刃有余。这也是UiBot在产品设计和技术实现上足够领先的资本。

1.3 UiBot发展历程

前面提到，在财务、物流、销售、人力资源等很多领域，都存在大量简单重复的软件操作，甚至到了摧残人性的地步。但实际上，早在上个世纪末，这种摧残人性的软件已经在游戏领域大量出现了。游戏其实也是一个需要人来完成的流程，但是，很多游戏开发者的设计水平不够，又希望玩家能在游戏中停留尽可能多的时间，所以故意把简单的流程重复无数遍，玩家苦不堪言。

于是，针对游戏领域的“软件机器人”应运而生，其中最著名的是2001年问世的“按键精灵”。按键精灵最早在Windows电脑上运行，针对Windows客户端游戏进行自动化操作；从2009年起，出现了“网页版按键精灵”，针对网页游戏进行自动化操作；从2013年起，又出现了“手机版按键精灵”，针对Android手机上的游戏进行自动化操作。这样一套产品体系，把主流游戏一网打尽，其技术积累之深厚可见一斑。

但按键精灵的成功决不在于技术上的优势，而是其“简单易用”的设计理念。按键精灵本身不是一个软件机器人，而是软件机器人的制造工具，这套工具要足够容易上手，让不是IT专家的游戏玩家也能轻松掌握，才算是达到“及格线”。在这一点上，按键精灵做得很成功，目前已经有几万名游戏玩家能够熟练的用按键精灵制造自己的“软件机器人”，并分享给更多的人使用，而这些玩家大多数并不精通IT技术，甚至没有接受过高等教育。

从某种意义上讲，2001年出品的“按键精灵”完全可以看作是国内RPA的先驱。实际上，当2017年RPA的概念在国内开始生根发芽的时候，国内有很多介绍RPA的文章，都会用按键精灵来举例子。虽然按键精灵本身是针对游戏设计的，和财务等领域的“软件机器人”有所不同，但因为名气大，容易理解，用来阐述RPA的概念再合适不过了。

那么，按键精灵的制作团队现在在做什么呢？他们在RPA方面有无斩获呢？当然有，他们认真分析了RPA的具体需求，对按键精灵进行了一次几乎推倒重来的大革新，既保留了团队十几年以来的积累，又积极满足RPA的需求，打造出一款强大、易用、快捷的RPA平台。没错，这就是UiBot！

现在，您终于明白为什么UiBot有资格傲视群雄了吧？

由于面向的领域不同，按键精灵和UiBot从基本理念上有很多不同点，技术上的差异更是天翻地覆：

- 按键精灵针对个人用户的需求做了很多优化，能制作用户界面，能设定热键，甚至能支持手机App的自动化操作，这些功能在UiBot中都被删掉了；
- UiBot针对企业用户做了很多优化，支持SAP自动化操作，能以流程图的方式展现自动化流程，支持分布式的控制中心，这些都是按键精灵不具备的；
- 按键精灵的主要指标是运行速度快，因为游戏画面瞬息万变，慢了会跟不上游戏的节奏；软件体积小，因为个人用户的下载带宽有限，这些指标在UiBot中并不重要；
- UiBot的主要指标是运行稳定性好，容错性强，遇到特殊状况宁可停下来，也不盲目操作，另外每次运行都有迹可循，这些指标都远远超过了按键精灵。

所以，到底用按键精灵还是UiBot，要看您的具体需求：如果是游戏领域，推荐您仍然使用按键精灵；如果是RPA，果断选择UiBot。

1.4 RPA和AI

RPA可以帮我们做很多事情，凡是在计算机上进行的简单、重复的操作，都可以考虑使用RPA来代替。但是，有一些重复而不那么简单的操作，是RPA无能为力的，仍然不得不由人工来完成。比如，我们可以依靠RPA，自动收邮件、回邮件，但它只能回复固定内容的邮件。如果要判断对方的意图，根据意图回复不同的邮件，RPA就做不到了。

人工智能（AI）在近年来得到了飞速的发展。比如市面上很多手机App都支持依靠人脸识别来进行身份认证，或者以对话的方式向手机发号施令，等等。这些都是人工智能的经典使用场景。那么，我们能不能把AI和RPA相结合，让AI来识别邮件内容的大致意图，然后告诉RPA，再让RPA去根据不同的意图，回复不同内容的邮件呢？

答案是肯定的！这个视频就生动地向我们演示了这一场景。它甚至更进一步，除了收邮件和回邮件之外，还让RPA和AI一起做更多、更有趣的事情。

随着RPA的发展，大家越来越深入地认识到：RPA本身也能做很多工作，但它是“无脑”的，如果和AI配合起来，RPA模拟人的双手，让AI模拟人的大脑。两者合力形成的“软件机器人”，就像下图所展示的一样，有脑有手，应用场景自然更加广阔。

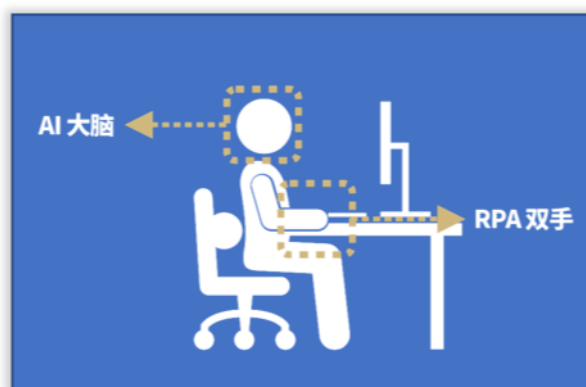


图 2: 软件机器人的双手和大脑

和按键精灵相比，UiBot也加入了大量AI的功能，从识别发票、身份证，到文本信息分类，文本内容抽取等，都是UiBot的拿手好戏。在后续章节中，我们会逐一教大家怎么用这些AI功能。是不是想想还有些小激动呢！

1.5 后续内容

前面提到，一般的RPA平台至少会包含三个组成部分：开发工具、运行工具和控制中心。

UiBot也不例外，在UiBot中，这三个组成部分分别被命名为UiBot Creator、UiBot Worker和UiBot Commander。

和一般的RPA平台相比，UiBot中还提供了专门为RPA设计的AI能力，这些AI能力也构成了UiBot的第四个组成部分，称之为UiBot Mage。

UiBot的四个组成部分及其关系如图所示：



图 3: UiBot的四个组成部分

如果只需要少量的电脑运行流程，可以由UiBot Creator制作出流程后，直接打包分发给UiBot Worker使用，UiBot Commander不需要参与；如果需要大量的电脑运行流程，比较合适的方式是UiBot Creator把流程先上传到UiBot Commander，再由UiBot Commander统一下发给各个UiBot Worker，并统一指挥它们运行流程。

当然，本文是UiBot的开发者指南，所以，本文的主要内容是介绍如何使用UiBot Creator去创建流程，以及如何使用UiBot Mage提供的AI能力，完成更多更有趣的任务。另外，也会用一章的篇幅介绍如何使用UiBot Worker及UiBot Commander去管理和运行流程。

在阅读本文的同时，建议您到北京来也网络科技有限公司的官方网站 <http://www.laiye.com> 或 UiBot的官方网站 <http://www.uibot.com.cn> 下载并安装一份UiBot的社区版，社区版是永久免费的，并且包含了UiBot的四个组成部分，只需要您在线登录一下，即可无限制地使用。

本文以2021年末发布的UiBot 6.0社区版为蓝本进行讲述。如果您还在使用老版本，可能会有很多界面选项和操作都和本文所述的有所不同，建议下载老版本的开发者指南。

如果您之前有一点点的编程经验，无论什么编程语言，只要知道什么是变量，什么是条件判断，那么阅读本文都会更加顺畅。如果没有，也没关系，我们在本教程最后一章 附录：编程基础知识 中，对UiBot涉及到的编程基础概念进行了简单的介绍，通俗易懂，请您在阅读后续内容之前，提前学习一下。

准备好了吗？下面将开始我们的UiBot之旅，Let's Go!

2 基本概念

是不是已经开始跃跃欲试，就想赶快装上UiBot软件，实现您的流程自动化了？

别急，别急，磨刀不误砍柴功。本章要介绍UiBot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，所以请务必牢记。这一章的内容不会太长，请务必耐心，不要轻易跳过哦！

2.1 概念

我们先来看下这四个基本概念。这几个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**。所谓流程，是指要用UiBot来完成的一项任务，一个任务对应一个流程。虽然可以用UiBot陆续建立多个流程，但同一时刻，只能编写和运行一个流程。将来在使用UiBot Worker和UiBot Commander的时候，也是以流程为基本单元来使用的。

如果您之前用过按键精灵，那么“流程”大致相当于按键精灵中的“脚本”。当然，UiBot中的“流程”和按键精灵中的“脚本”又有一定的差异，比如“流程”包含一个文件夹，而不只是一个文件。更重要的差异是：UiBot中的流程，都是采用流程图的方式来显示的。

俗话说得好，“纸上得来终觉浅，绝知此事要躬行”，UiBot是一种实践性很强的工具，所以我们建议在学习本教程的同时，打开UiBot软件，亲自将里面的内容实践一下，相信学到的知识会更加深刻。

UiBot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。

2.2 流程图

新建或打开UiBot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，包含了一系列的“组件”，其中最常用的是“开始”、“流程块”、“判断”和“结束”这四种组件，其他的如“辅助流程”、“子流程”对于初学者来说可以先不掌握，留待后续内容中再学习。

用鼠标把一个组件从左边的“组件区”拖到中间空白的“画布”上，即可新建一个组件，如新建一个流程块等。在画布上的组件边缘上拖动鼠标（此时鼠标的形状会变成一个十字型），可以为组件之间设置箭头连接。

把多个组件放在一张画布上，用箭头把它们连起来，则构成一张流程图。如下图所示：



图 4: UiBot 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“是”箭头运行后续组件；否则，沿着“否”箭头运行后续组件。如果您是UiBot的新手，可能暂时还用不到“判断”组件。本章后续部分会详细叙述“判断”组件。

最后，也是最重要的，流程图中必须有一个或多个“流程块”，**流程块**是本章要介绍的第二个重要概念。

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UiBot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，这个例子只是以玩笑的方式打个比方，UiBot并不能帮我们把冰箱门打开。但通过这个例子可以看出，在UiBot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，而暂时不涉及到如何去做的细节。

UiBot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，在一个流程中，建议不要超过20个流程块。

为什么这样建议呢？其实UiBot并未规定流程块的数量上限，如果您愿意，放200个、2000个流程块都没有问题。但是，UiBot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UiBot的工具栏上，有一个“运行”按钮。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三角形状的按钮，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。

2.3 可视化视图

上一节提到，每个流程块上还有一个形状类似于“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，UiBot的界面会从“流程视图”切换到“可视化视图”。



图 5: 流程图中点击编辑流程块

UiBot编写流程块的“可视化视图”，其界面如下图所示：

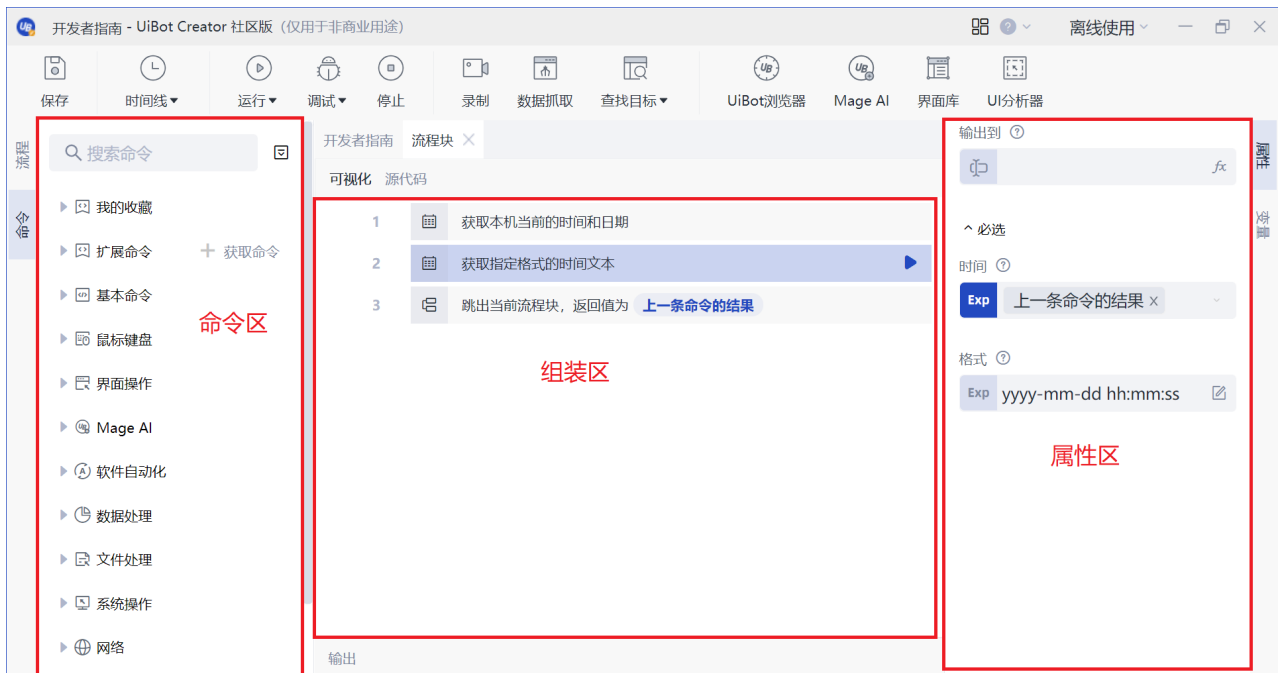


图 6: 流程块编辑界面(可视化视图)

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、属性区。

这里引入第三个重要概念：**命令**。所谓命令，是指在一个流程块当中，需要告知UiBot具体每一步该做什么动作、如何去做。UiBot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，UiBot并不能把冰箱门打开。UiBot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。

命令是我们要求UiBot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，UiBot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，UiBot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UiBot作为RPA平台的“简单”这一重要特点，为此，UiBot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。

将命令区、组装区、属性区从左到右进行排列，是UiBot的默认排列方式。您也可以拖动每个区域上的标签，将其调整到您喜欢的排列方式。

2.4 源代码视图

您也许已经注意到了，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，此时，属性变量区会消失，组装区会变成如下图所示的样子：



图 7: 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UiBot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。UiBot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，让您以快意的心情书写一条条的命令。



图 8: 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。



图 9: 飞机的两种视图

打个比方，可视化视图和源代码视图就像是上面这张飞机的视图一样。其实这架F-16飞机的左右两翼是基本对称的，但为什么看起来不一样呢？因为它的右翼采用外观视图绘制，展现整体造型，左翼采用透视视图绘制，展现内部构造。同一架飞机，用两种视图展现不同的内容，才能兼顾不同观众的关注点。同样道理，同一个流程块，用两种视图展现不同的内容，才能兼顾RPA平台的“简单”和“快捷”两大

指标。

有的读者会问，究竟我要使用可视化视图，还是源代码视图来进行RPA流程开发呢？其实，您大可不必纠结于此，因为无论您使用哪种视图，都可以随时切换到另一种视图。您在一种视图上无论编写了什么内容，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。所以，您完全可以先用可视化视图，稍微熟悉一点儿以后，切换到源代码视图尝尝鲜，也了解一下内部原理，如果觉得暂时还有困难，再切换回可视化视图就好了。完全没有选择恐惧症！这也是UiBot的强大之处！

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UiBot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UiBot自研的BotScript语言，具体的语言特性，将在后文详细描述。

2.5 小结

我们在这一章学习了四个重要概念：流程、流程块、命令、属性。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。我们还看到了三种视图：流程视图、可视化视图、源代码视图。流程视图是流程的展现，可视化视图和源代码视图都是流程块的展现。它们之间的关系如下图所示：

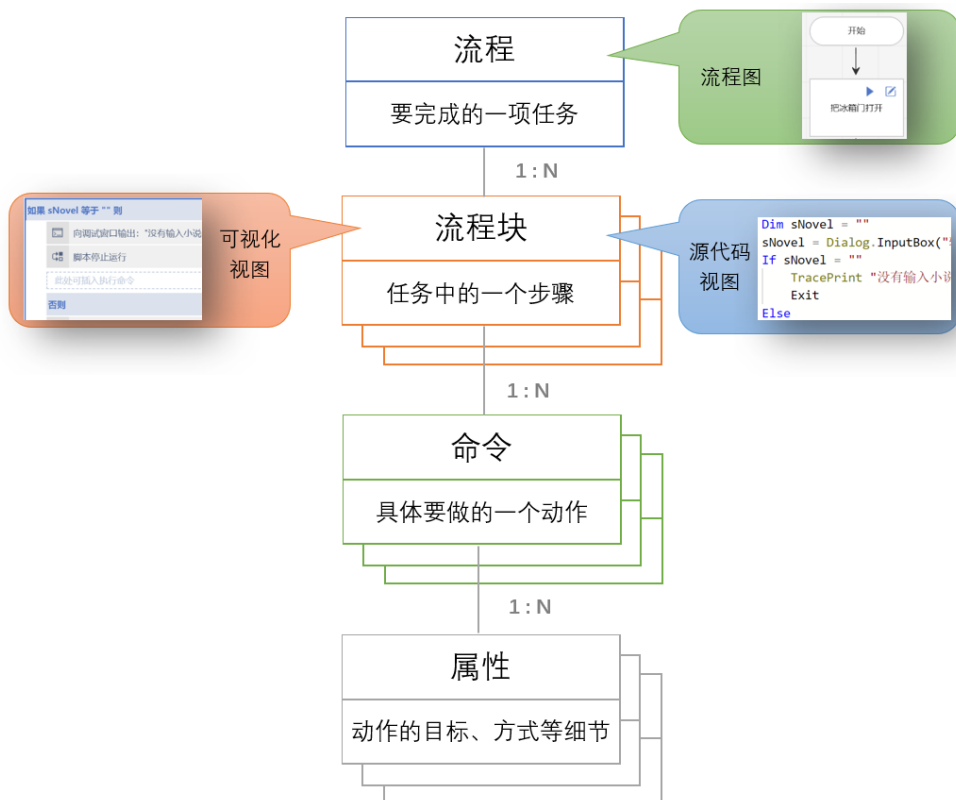


图 10: 四个概念和三个视图的关系

2.6 进阶内容

本节是进阶内容，当您在多个流程块之间共享和传递数据、在流程图中使用“判断”组件的时候，请阅读本节内容。如果您是UiBot的初学者，可以跳过不读。

2.6.1 流程图变量

流程图和流程块中都可以使用“变量”来存储数据：流程块中的变量，使用范围仅限于当前流程块中，在流程图和其它流程块中无法直接使用；从UiBot Creator 5.0版本开始，在流程图中也可以定义变量。如果在流程图中定义了一个变量，那么在流程图所包含的所有流程块中，都可以直接使用这个变量。

下面，我们将举例说明流程图变量的具体用法。

假设有一张流程图，包含两个流程块，分别命名为“流程块1”和“流程块2”，如下图所示。“流程块1”先运行，它的功能是获得当前系统时间，并将系统时间转换为字符串格式。“流程块2”后运行，它的功能是把“流程块1”生成的字符串格式的系统时间，以调试信息的方式显示出来。

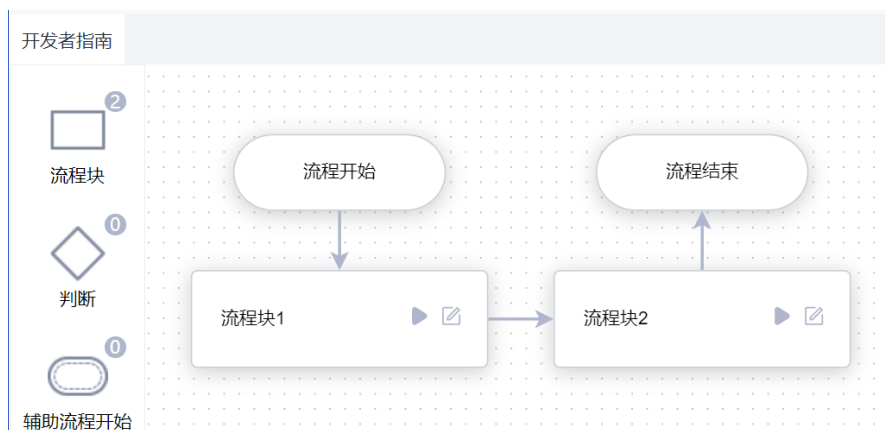


图 11: 两个依次运行的流程块

由于“流程块1”和“流程块2”之间需要传递“字符串格式的系统时间”这一数据，可以将其保存在流程图变量中，进行传递。我们首先在流程图中定义这个变量。在“流程图”视图，找到并点击位于右侧的“变量”标签，可以看到所有的流程图变量，点击“添加”按钮，输入变量名x（不区分大小写），即可增加一个流程图变量。

每个流程图变量还可以指定“使用方向”，包括“输入”、“输出”、“无”这三种使用方向。其中“输入”和“输出”都是高阶功能，在子流程中才需要使用的。我们只考虑当前这一个流程的话，将使用方向设为“无”就好。



图 12: 在流程图视图添加变量

点击流程块1的“纸和笔”图标进入流程块1的可视化视图，插入一条“获取本机当前时间”和一条“获取指定格式的时间文本”命令（在“时间”分类下），并把“获取指定格式的时间文本”中的“时间”属性设为“获取系统时间”的结果，即可得到当前时间，并以容易阅读的字符串格式保存在流程图变量x中。由于x是流程图变量，因此，在下一个流程块中，可以直接使用x的值。

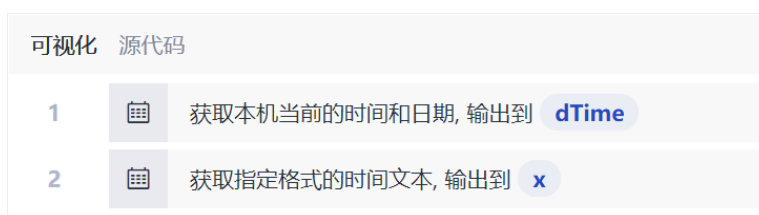


图 13: 流程块1的实现

如果觉得输入变量名称太麻烦，或者变量名太长记不住。也可以在流程块的“可视化视图”的“属性”区域，找到一个标有字母fx的按钮或者下拉按钮，按下后，会弹出一个菜单，列出所有可用的流程块变量、流程图变量和系统变量，用鼠标点选即可。如上所述：“流程块变量”仅限于当前流程块使用，“流程图变量”可以在整个流程图的任意一个流程块中使用，此外还有“系统变量”，是您不需要定义也不需要赋值的，可以直接使用里面预置的值。



图 14: 在下拉菜单中选择变量

下面，再点击“纸和笔”图标进入流程块2，插入一条“输出调试信息”命令，并把“输出内容”属性设为x（如前所述，变量x为流程图变量，可以直接使用，而不需要再定义）。

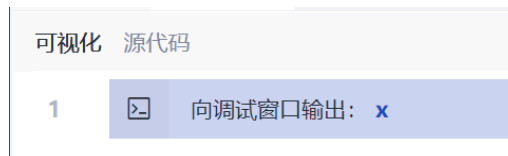


图 15: 流程块2的实现

回到流程图界面，然后点击“运行”按钮，即可看到运行结果，显示出当前时间。

2.6.2 流程块的输入输出

上一节展示了用流程图变量在两个流程块之间传递数据。这种方式简单易懂，但略有些麻烦：明明是两个流程块之间传递数据，非要把流程图也牵扯进来，显得多此一举。其实，不依靠流程图变量，两个相邻的流程块之间，也可以传递数据。前一个流程块运行结束的时候，可以将一个值“输出”，这个输出值通过两个流程块之间连接的箭头，直接“传导”到下一个流程块中。

那么，如何在前一个流程块里面输出一个值，并且在后一个流程块里面获得这个值呢？

很简单，首先，进入前一个流程块，在“命令区”里面找到“词法语法”下面的“跳出返回”命令，将其拖拽到“组装区”，并在“属性区”里面设置其输出值，可以是数值、字符串，也可以是变量或表达式。或者，如果您习惯使用源代码视图，在流程中书写`Return <输出值>`，效果也是一样的。在运行的时候，遇到这条命令，就会跳出当前流程块，并且把输出值传到后一个流程块。

下面，进入后一个流程块，当其中有某条命令需要使用前一个流程块的输出值（也就是当前流程块的输入）的时候，直接在“属性区”里面，找到相应的属性，点击右侧标有字母fx的按钮，并在弹出的菜单中选择“系统变量”中的“流程块的输入”（如下图所示），即可在运行的时候自动获得这个值。当然，如果您习惯使用源代码视图，也可以直接书写`$BlockInput`（变量前面的\$符号表示这是一个系统变量）。



图 16: 选择系统变量“流程块的输入”

下面，我们仍将用上一节的例子说明在流程块之间直接输出和输入的具体方法。

首先进入“流程块1”，它的功能是获得当前系统时间，并将系统时间转换为字符串格式。假设转换后的结果保存在变量sRet中，那么下面只需要拖入一条“跳出返回”命令，并把sRet作为其输出值即可。这里还有一个技巧：当“获取指定格式的时间文本”命令和“跳出返回”命令相邻的时候，甚至连变量sRet都不需

要，直接在“跳出返回”的属性里面按fx按钮，并选择系统变量“上一条命令的结果”即可，这样可以省去一个变量，并且效果也是一样的。



图 17: 选择系统变量“上一条命令的结果”

然后进入“流程块2”，插入一条“输出调试信息”命令，并把“输出内容”属性设为系统变量“流程块的输入”。运行整个流程图，可以看到，也得到了预期的结果。当然，这里不能只运行流程块2，否则，由于缺乏流程块1的输出，结果就会不正确了。

采用流程图变量，或者使用流程块之间的输出和输入，是流程块之间传递数据的两种方式。前者使用起来更加直观，且当流程块之间的箭头发生变化时，也不会影响流程图变量的值。后者依赖于两个次序相邻的流程块，逻辑上更加清晰一些。采用哪种方式，取决于读者的习惯。

2.6.3 复杂流程图的实现

有人可能会有这样的疑惑，UiBot只提供了“开始”、“流程块”、“选择”和“结束”四种组件，其中“开始”和“结束”还不是真正的业务流程组件，仅仅通过“流程块”和“选择”这两种组件，能够胜任复杂的流程图吗？就好比下图：

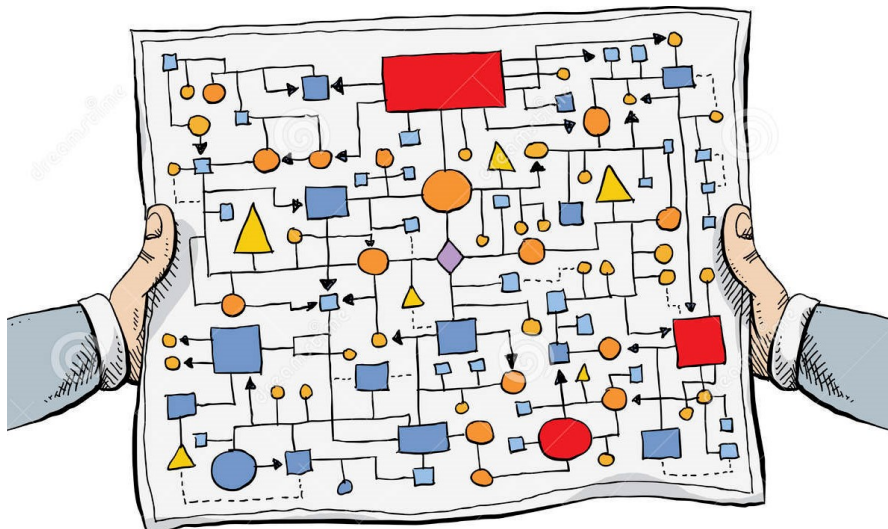


图 18: 复杂流程图

您还真别小看UiBot这几种组件，再复杂的流程图，也可以通过这四种组件的简单排列组合来实现！

我们来透过现象看本质：再复杂的流程图，按照其结构组成来分类，大致可以分为三种：顺序结构、选择结构和循环结构，下面我们就分别来看看这三类流程图如何用UiBot来实现。

顺序结构

在顺序结构中，各个步骤是按先后顺序执行的，这是一种最简单的基本结构。如图所示，A、B、C是三个连续的步骤，它们是按顺序执行的，即完成上一个框中指定的操作才能再执行下一个动作。

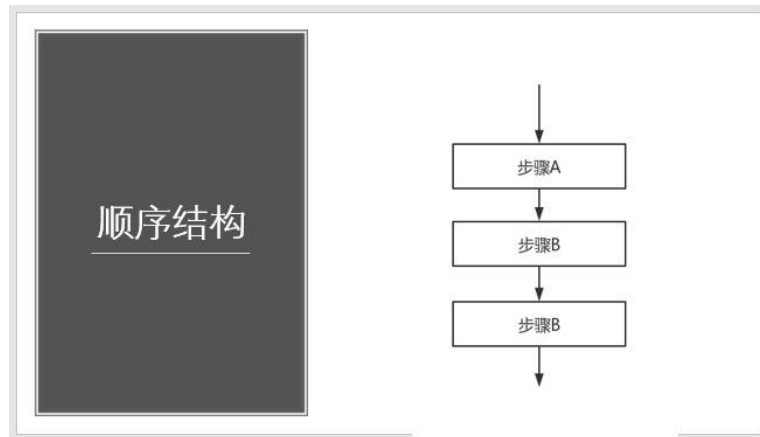


图 19: 顺序结构

UiBot中实现顺序结构的顺序流程图如图所示。



图 20: 顺序流程图

选择结构

选择结构又称分支结构，选择结构根据某些条件来判断结果，根据判断结果来控制程序的流程。在实际运用中，某一条分支路线可以为空（如图二、图三）。

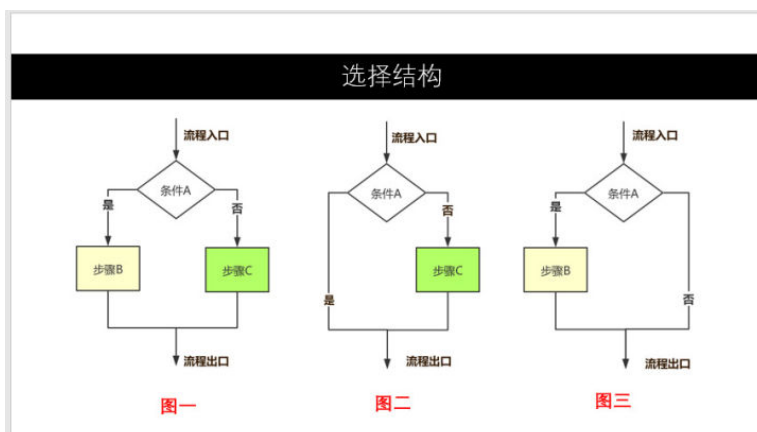


图 21: 选择结构

UiBot中实现选择结构的选择流程图如图所示。

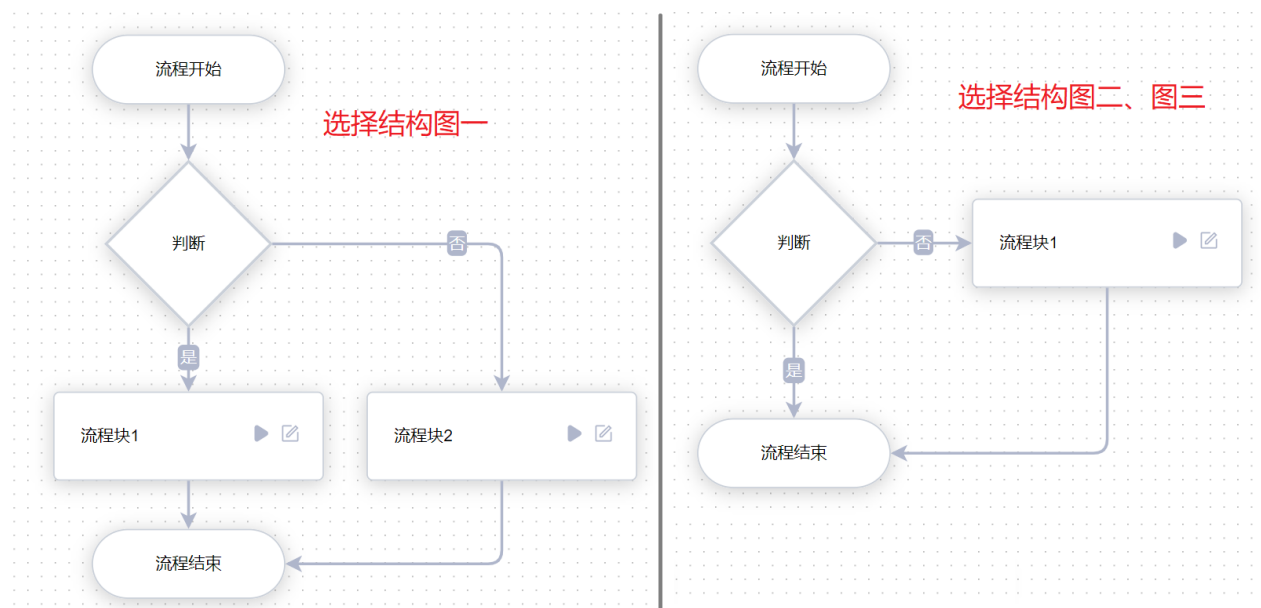


图 22: 用UiBot实现选择结构图一、图二和图三

循环结构

循环结构又称为重复结构，指的是流程在一定的条件下，反复执行某一操作的流程结构。循环结构下又可以分为当型结构和直到型结构。

循环结构可以看成是一个条件判断和一个向回转向的组合，使用流程图表示时，判断框内写上条件，两个出口分别对应着条件成立和条件不成立时的执行路径，其中一条路径要回到条件判断本身。

当型结构：先判断所给条件P是否成立，若P成立，则执行A（步骤）；再判断条件P是否成立；若P成立，则又执行A，若此反复，直到某一次条件P不成立时为止。流程结束。

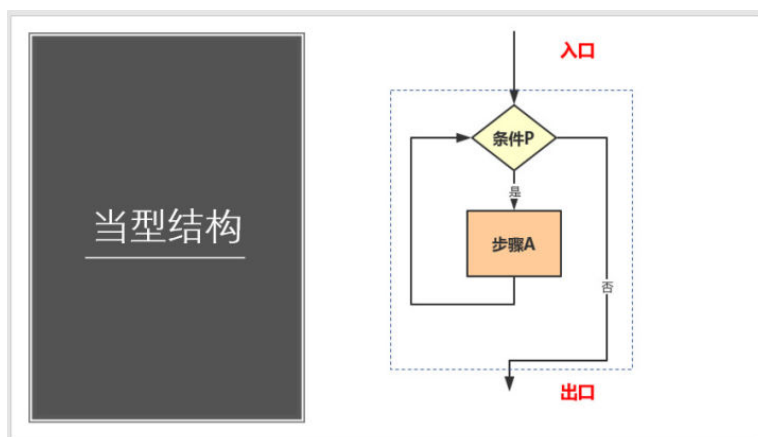


图 23: 当型结构

直到型结构：先执行A，再判断所给条件P是否成立，若p不成立，则再执行A，如此反复，直到P成立，该循环过程结束。

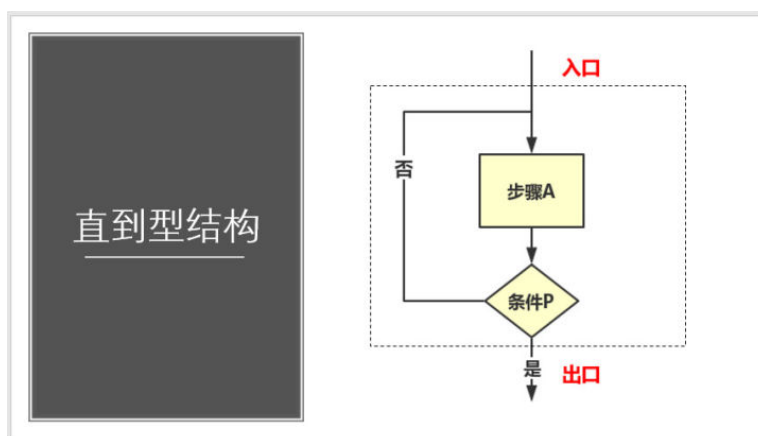


图 24: 直到型结构

UiBot中实现循环结构的循环流程图如图所示。

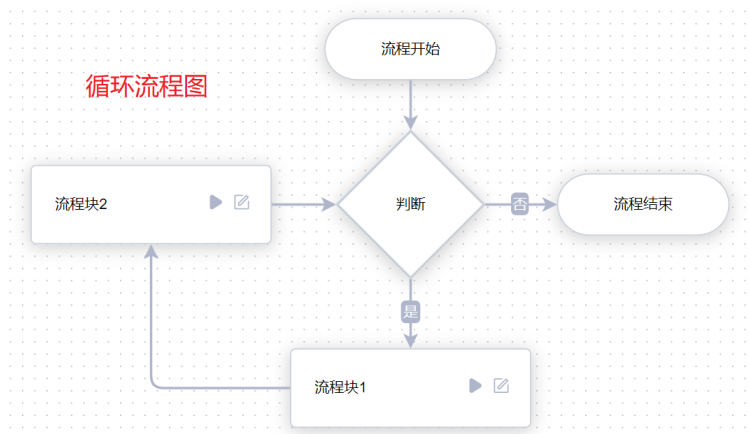


图 25: 用UiBot实现循环流程图

具体使用上，UiBot使用“判断”组件来实现上述功能，把“判断”组件拖到流程图中，并且用鼠标左键选中，即可在属性栏中看到该组件的属性。如图所示。其中“条件表达式”这一栏很关键，您可以填写一个变量或者表达式（这里只能使用流程图变量）。在流程运行到此判断时，将根据这个变量或表达式的值是否为真，来决定后面沿着“是”所示的出箭头继续运行，还是沿着“否”所示的出箭头继续运行。

描述

判断

条件表达式

$x > 1$

图 26: 判断表达式

“判断”组件有两个出箭头，一个标有“是”，一个标有“否”，当其属性中的“条件表达式”为真时，沿着“是”箭头往后运行，否则，沿着“否”箭头往后运行。如图所示。

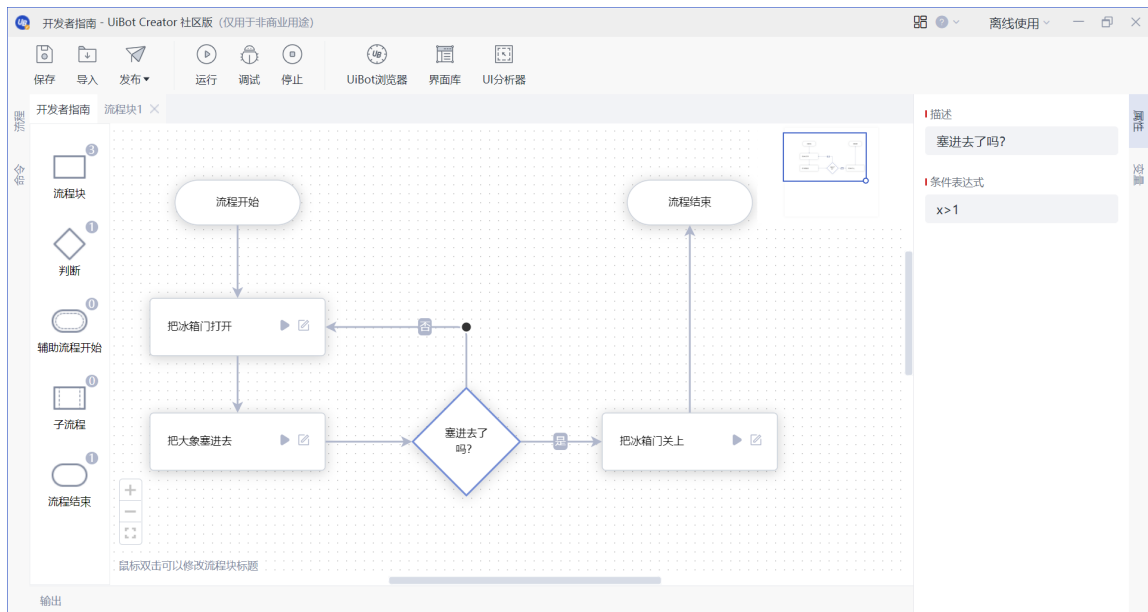


图 27: 根据条件表达式来决定流程

3 界面元素自动化

我们在“RPA简介”一章中曾提到，RPA的一大特色是“无侵入”，也就是说，虽然RPA是配合其他软件一起工作的，但并不需要其他软件提供接口。而是直接针对其他软件的操作界面，模拟人的阅读和操作。但是，一般的软件界面上都会有多个输入框、按钮，计算机怎么知道我们到底要操作什么地方呢？本章所述的“界面元素”和“界面元素自动化”将解决这个问题。

3.1 界面元素

如果您之前有一定的计算机基础，了解什么是“控件”，对不起，请先暂时忘掉这个概念。因为“控件”和“界面元素”虽然有共同点，但又不完全一样，一定要尽量避免概念混淆。

除了计算机专家之外，一般人在使用计算机的时候，都是在和操作系统的图形界面打交道。无论是常用的Windows或Mac OS X，还是非IT人士不太常用的Linux，都有一套自己的图形界面。随着Web浏览器的大行其道，也有越来越多的图形界面选择在浏览器上展现。这些图形界面各有各的特色，但当我们用鼠标点击的时候，其实鼠标下面都是一个小的图形部件，我们把这些图形部件称为“界面元素”。

比如，下图是一个普通的Windows窗口，也是典型的图形用户界面。在这个窗口中，有哪些界面元素呢？

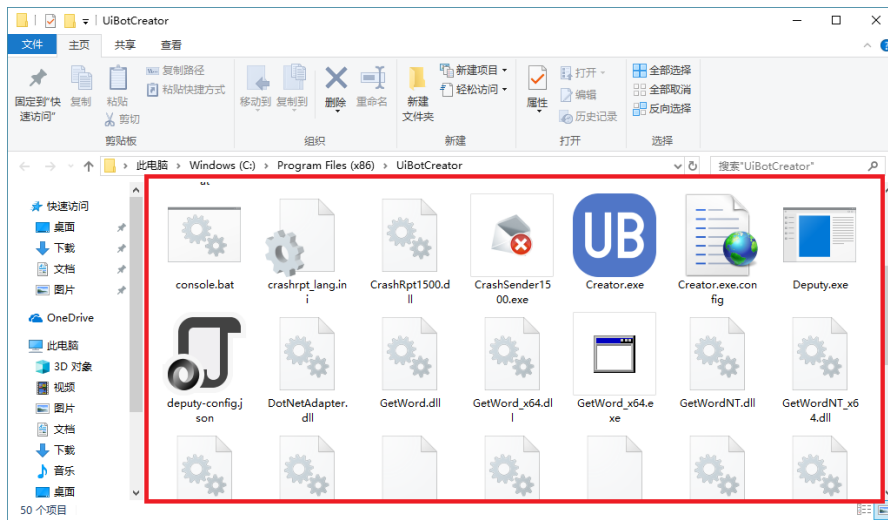


图 28: 普通的Windows窗口

首先，上面的菜单栏里面的各个选项，如“文件”、“主页”、“共享”、“查看”都是独立的界面元素。菜单栏里面的图标和下面的文字，如“复制”、“粘贴”等都是独立的界面元素，左边的导航栏里面的“快速访问”、“桌面”、“下载”等都是独立的界面元素；当然，窗口主要区域（红框包含的范围）里面显示的每个文件也都是独立的界面元素。

界面元素之间还有嵌套的组合关系。比如，红框包含的范围是一个大的界面元素，里面的每个文件又是独立的界面元素。

在UiBot中，界面元素的作用，就是作为“有目标”的命令中的目标使用。

前文中提到，UiBot在命令区已经放置了很多作为“预制件”使用的命令。其中，最基础的是如图所示的几类：



图 29: 最基础的几类命令

其中，“界面元素”和“文本”类别下面的所有命令，都是有目标的；“鼠标”和“键盘”下面的包含“目标”两个字的命令，也都是有目标的。如下图红框所示。



图 30: 有目标的命令

所谓有目标的命令，就是在命令中指定了一个界面元素。在运行的时候，会先查找这个界面元素是否存在。如果存在，则操作会针对这个界面元素进行。比如界面元素是一个按钮，那么命令“点击目标”就是点击这个按钮。如果不存在，则会反复查找，直到超过指定的时间（也称为“超时”。超时时间可以在“属性”中设置），会输出一个出错信息，流程也会直接停止运行。

相反，对于无目标的命令，在命令中就不需要指定界面元素了。比如“模拟点击”命令是没有目标的，在运行的时候，鼠标当前在什么位置，就点击什么位置。究竟点了什么东西，是无法验证的；再比如“模拟按键”命令也没有目标，在运行的时候，键盘的输入焦点在什么位置，就在什么位置模拟一个按键操作。到底输入到哪里去了，也无法验证。

显然，在用UiBot的时候，应该优先使用有目标的命令，因为有目标的命令指定了操作的对象，会比较精准。只有当找不到目标的时候，才退而求其次，使用无目标的命令。

所以，在用UiBot的时候，如何准确选取一个目标是很关键的。只要准确地选到了目标，模拟操作相对来说就比较简单了。下面介绍选取目标的方法。

3.2 目标选取

UiBot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”视图的“命令区”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。以上步骤您应该已经很熟悉了，如果还不熟悉，请回过头去阅读基本概念这一章。

在组装区中，现在已经有一条命令了。我们会注意到，这条命令上有一个瞄准器样子的图标，其右侧还有文字“未指定”，这两者的含义实际上是“目标还没有指定”。鼠标移动到这个图标上，自动弹出一个菜单，其中有两个选项：“从界面上选取”和“从界面库选择”。如下图所示：



图 31: “点击目标”命令和“未指定目标”按钮

点击“从界面上选取”，UiBot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，UiBot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。

前文提到，界面元素可能是嵌套的，鼠标所在的位置，可能已经落到了多个界面元素的范围之内。此时，目标选择器会自动选择您最有可能需要的界面元素，并将其遮住。所以，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。在目标选择器的上方，UiBot还会显示出这个界面元素的类型，比如是一个按钮，还是一个输入框，等等，这些信息也是为了帮助您判断是否恰好选择了您需要的界面元素。因为很多比较复杂的界面，往往都会有多个位置非常接近的界面元素叠放在一起，而它们当中可能只有一个是您真正希望当作“目标”的，所以要仔细辨认，不要搞错了。

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。



图 32: 用“目标选择器”选中开始菜单按钮

一旦选中作为目标的界面元素之后，UiBot的界面重新出现，刚才的那条“鼠标按下”的命令，其中的瞄准器已经变成了这个界面元素的名字（名字是UiBot自动取的，稍后您会学习到如何改这个名字）。把鼠标移上去，还会有一个浮窗显示出界面元素的缩略图。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。缩略图的下面还有一个“重新从界面上选取”的按钮，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。

与此同时，这个被选取的界面元素，UiBot也会把它自动保存到一个“界面库”里。如果在**同一个流程**的任何一个流程块中，还需要再次使用到这个界面元素，就不必费力再选取一次了。在命令上找到“从界面库选择”，点击后，直接在界面库里保存的多个界面元素中，选择您需要的那一个即可。注意：每个流程都会有自己专属的“界面库”，同一个流程的多个流程块中，可以共享同一个界面库。而在流程与流程之间，界面库就不会共享了。



图 33: 在“界面库”中直接选择之前保存的界面元素

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，UiBot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UiBot，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UiBot中，您完全无需关心这些区别，UiBot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。

3.3 目标编辑

在上一节中，我们看到UiBot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，UiBot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。我们先来解释为什么会发生这种情况，然后再来看如何解决。

下面请各位读者来试一下，点击这里打开一张表格。这个表格是笔者虚构的，它是一张“员工信息表”，其中有五个虚构的员工，在表格的第三列，也就是“SALARY”这一列，显示了员工的薪资（当然，也是虚构的）。如下图所示：

员工信息表

ID	NAME	SALARY	EMAIL
1	朱元璋	6900	zhuyz@ming.com
2	赵匡胤	6700	zhaoky@song.com
3	李世民	6800	lism@tang.com
4	刘邦	5200	liub@han.com
5	嬴政	7600	yingz@qin.com

图 34: 虚拟的“员工信息表”

我们可以试着用UiBot来找到第三名员工，也就是“李世民”的薪资所在的单元格（在上图中用红框标出。单元格里面的数组是随机生成的，可能和您的浏览器里面显示的数值不同），并且让UiBot自动控制鼠标去双击这个单元格。

这个任务实在太简单了！在UiBot里新建并打开一个流程块，在“鼠标”分类下面拖入一条“点击目标”的命令，在属性栏里把“点击类型”设为“双击”，其他属性保持默认值不变。然后，用“从界面上选取”的功能，选择“李世民”那一行，“SALARY”那一列的单元格，作为命令的目标，这个任务就完成了。运行流程，可以看到鼠标自动双击了目标的单元格，单元格里面的文字也因为双击而变成了选中的状态。

但是，千万别高兴得太早。试着刷新一下这个网页，再运行刚刚编写好的流程，这次，流程的运行会失败，报告找不到界面元素。界面元素明明存在，但却找不到它，这就是发生了“漏选”。

这是为什么呢？

我们稍微留意一下，就会发现在刚才拖入的命令的属性中，有一条属性被称为“目标”。当我们还没有选择目标的时候，这个属性的内容是“未指定”。而当选择了目标以后，这个属性的内容会变成作为目标的界面元素的名字。当目标明明存在，但就是找不到的时候，显然是这里的“目标”出了问题。不难发现，在这个属性的右侧，还有一个“纸和笔”形状按钮，我们称之为“编辑”按钮，如下图：

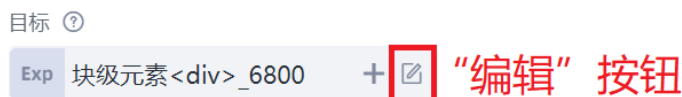


图 35: “目标”属性及其“编辑”按钮

点击“编辑”按钮，会弹出一个对话框，展示这个作为目标的界面元素的一些细节，如下图所示：

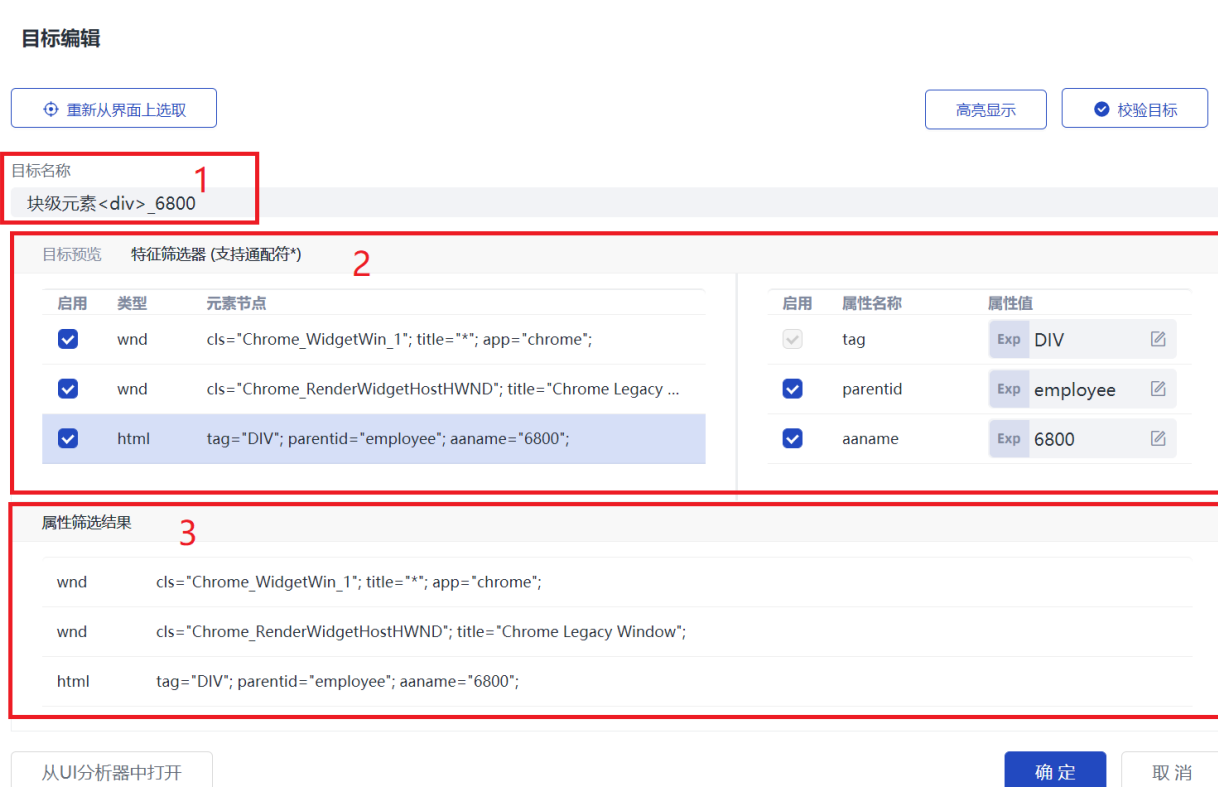


图 36: “目标编辑”对话框

这个对话框里的内容，您第一次看到的时候，可能会有些发懵，因为它看起来有些复杂。稍安勿躁，我们以庖丁解牛的方式来慢慢解析。

我们把这个对话框分为三个主要的区域，在上图中用区域1、2、3来表示，除了这三个区域之外，剩下的一些按钮都很容易顾名思义。

区域1主要是这个界面元素的名字，前面提到过，这个名字是UiBot自动起的，您可以在这里修改它。

区域3描述的实际上是这个界面的一些“特征”，这个特征分了好几组，每一组里又会包含一系列的属性和值。比如上图中，位于第三行的组名叫html，其中又包含了三个属性，分别是tag, parentid, aaname，这些属性的值依次是“DIV”，“employee”和“6800”。UiBot在判断一个界面元素是否存在的时候，会查找其中所有组里的所有属性和值，只有当它们全部匹配的时候，才认为这个界面元素是存在的。

实际上，UiBot在描述一个界面元素的特征的时候，还有很多的备选项，它会自动从这些备选项里面，找出它认为比较合适的组，以及其中相应的属性和值。在区域2里面，UiBot会把所有的备选项都逐一列出，如果前面打勾，表示某个组或某个属性会被选入界面元素的特征。这里的值还可以修改为任意您希望的值，甚至可以是变量或者表达式，也可以用星号通配符（即 * 符号）来表示模糊匹配任意长度的字符串。

您可能会注意到，这里有个叫aaname的属性，其值是“6800”。它是刚才造成“漏选”的“元凶”。因为大致可以猜到，这个属性代表了单元格的文字内容，把文字内容当作界面元素的特征中的一部分，固然有助于找到界面元素，但上面的这个“员工信息表”比较特殊，每次刷新，它的内容都会发生变化。如果仍然把aaname的属性和“6800”这个值作为特征的一部分，就会像“刻舟求剑”一样，表格内容稍微发生一些变化就找不到了。

那怎么办呢？通常的做法是：对于这种可能造成“漏选”的属性，我们要把它从界面元素的特征中剔除。有两种方法可以剔除这个属性，一种是把属性前面的选择框的勾选状态取消，另一种是把属性的值改为星号通配符（即 * 符号）。如下图所示：



图 37: 剔除属性aaname的两种方法

我们可以实际尝试一下，用上述的任意一种方法，把aaname属性剔除掉，然后再运行这个流程。这次确实不会报错了，但却双击了第一行（也就是“朱元璋”那一行）的SALARY。这种情况，就是典型的“错选”。发生“错选”的原因是：这张“员工信息表”里面的五个员工，在SALARY这一列的界面元素特征除了aaname属性之外，其他全都一模一样。所以剔除了这个属性之后，UiBot会找到第一个符合这个特征的界面元素，也就是“朱元璋”这一行的SALARY列。

所以，当造成了“错选”的时候，还需要加入其他的属性来弥补，让UiBot避免“错选”。如何加入其他属性？我们会在后面介绍“UI分析器”的时候讲述。

总而言之，如果界面元素比较复杂，或者特征经常发生变化，如何准确地编辑目标，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、tag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aname属性的值是“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"姓名：*"来作为这个属性的值，以避免“错选”；
- 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：UiBot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，UiBot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Chrome和IE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Chrome和IE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。



图 38: 用Chrome和IE，同一目标的特征不同

所以，在用UiBot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。

3.4 界面元素操作

UiBot提供的界面元素操作的命令列表如下图所示。



图 39: 界面元素操作菜单

很多UiBot的初学者，在面对如此庞杂的命令列表时，往往显得不知所措。其实，这些命令大可不必死记硬背，只要熟练掌握一些技巧，深入理解UiBot的设计理念，就会发现其中存在一些通用的原理，掌握了这些原理，在面对特定问题时，脑中自然就会浮现出某条命令来，甚至这条命令应该有哪些属性、每个属性可以有哪些取值、每个取值的含义是什么，大概也能猜得出来，下面我们就一起来学习一下。

前文讲述过，对界面元素的操作通过命令进行，除了命令本身之外，命令的属性也是组成命令不可或缺的部分。甚至可以说，只有把命令的关键属性描述清楚，一条命令才能称之为一条完整的命令。一般来说，命令由如下几个部分组成：

命令 = (对什么事物) + (用什么东西) + (做什么操作) + (得到什么结果)

“对什么事物”，指的是命令的操作对象，也就是我们本节所说的目标，对于界面元素而言，操作对象包括单选框、多选框、文本框、列表框、下拉框等。前面我们已经学习了，可以通过UiBot的“从界面上选取”功能来确定命令的操作对象。

“做什么操作”，指的是对目标能够进行的操作。一般来说，对目标能够进行的操作是由目标本身的类型决定的，当目标确定了以后，能够对目标进行哪种操作，也就基本确定了。比如按钮，“点击”是能够进行的操作，而文本标签一般不能点击；再比如单选框，能够进行的操作，一种是获取它的选择状态（勾选还是没有勾选），一种是设置它的状态（勾选还是不勾选）。还有一些通用操作，是几乎所有界面元素都有的，比如获得界面元素的大小、位置、文字等等。

“用什么东西”，指的是对这个目标进行操作的时候，还需要用户提供哪些信息。这个同样取决于目标本身的类型。比如设置单选框的状态，传入的就应该是一个布尔值（True或者False）。也有少数命令，不需要用户额外提供信息，就可以执行，比如获取元素文本命令，只要告诉这条命令的目标，就可以获得元素文本，不需要其它信息。对于有编程经验的读者来说，“用什么东西”有点像编程里面的参数，不需

要额外提供信息则等同于编程里的**无参调用函数**。

“**得到什么结果**”，指的是有的命令还会有一个“输出值”。比如获取元素文本，输出的是字符串类型的元素文本；再比如**获取元素选择**命令，当界面元素是多选框时，输出的是一个数组（因为用户可能进行了多选）。对于有编程经验的读者来说，“输出值”相当于编程里面函数的“返回值”。

从上面的描述可以看得出来：本质上，UiBot是对真人操作的模拟，是将一个一个的操作封装成一条一条的命令，而这些命令的关键属性都是由目标决定的。

下面我们以一个具体的界面，来演示如何进行界面元素操作。这个界面中有文本标签、单选框、多选框、单行文本框、多行文本框、列表框、下拉框、勾选框等，基本代表了典型的界面元素。

WQM测试表单 (注册页面)	
用户名	<input type="text"/>
密码	<input type="password"/>
E-Mail地址	<input type="text"/>
性别	<input type="radio"/> 男 <input type="radio"/> 女
省份城市	省/市 <input type="text"/> 市/地区 <input type="text"/>
求职意向	选择职位类别 <input type="checkbox"/> 销售 <input type="checkbox"/> 市场/市场拓展/公关 <input type="checkbox"/> 商务/采购/贸易 <input type="checkbox"/> 计算机软、硬件/互联网/IT <input type="text"/> 按住Ctrl可以多选
爱好	<input type="checkbox"/> 音乐 <input type="checkbox"/> 运动 <input type="checkbox"/> 电影 <input type="checkbox"/> 购物
自我评价	<input type="text"/>
	<input type="checkbox"/> 我已阅读并接受注册协议
	<input type="button" value="提交"/>

图 40: 测试界面

3.4.1 判断元素是否存在

顾名思义，“判断元素是否存在”这一命令可以检查当前屏幕上是否出现了某个特定的界面元素，并且把检查的结果输出到一个变量里面。界面元素存在时，变量里面会保存True，否则保存False。

当我们需要判断流程执行到某个步骤后，是否会出现某个特定的界面，用这个命令是比较适合的。可以用界面上一个关键的元素作为判断标准，如果这个元素存在，表明出现了该界面，否则，表明该界面不存在。

例如，我们来判断上图的测试界面中是否存在“提交”元素。

1. 通过鼠标拖动，将“判断元素是否存在”加入到可视化视图中，如图所示；



图 41: 判断元素是否存在

2. 点击“判断元素是否存在”的“从界面上选取”按钮，切换到测试页面，将鼠标悬停在“提交”按钮上，使红边蓝框刚好覆盖“提交”按钮，点击鼠标左键；
3. 运行完成后，该命令的运行结果会输出到变量bRet中，如上图，我们通过一条判断语句，判断得到的结果是True还是False：True表明找到了界面元素；False表明没找到界面元素；

3.4.2 设置/获取元素勾选

“设置/获取元素勾选”用来自动化操作界面，自动完成表单填写、自动操作等功能，操作的元素主要是单选框和多选框。

我们同样以上图的测试界面为例，说明这两条命令的用法。

1. 通过鼠标拖动，将“设置元素勾选”命令加入到可视化视图中，“目标”为爱好的“音乐”选项和“运动”选项，注意需要查找的目标为音乐和运动的勾选项，而不是音乐和运动文字本身；
2. 通过鼠标拖动，将“获取元素勾选”命令加入到可视化视图中，“目标”同样为爱好的“音乐”选项和“运动”选项；
3. 运行流程，结果如图所示，输出内容为“爱好音乐”和“爱好运动”，说明成功设置元素的勾选；



图 42: 设置/获取元素勾选命令及运行结果

4. 在测试界面上，我们也可以看到，原来没有勾选“音乐”和“运动”选项，而现在这两个选项都已经自动勾选上了。因此，通过这两条命令，可以达到自动填写表单的目的。

3.4.3 获取子元素

这一节需要用到一点点编程语言里面“数组”的知识。如果您还没有掌握，可以参考后续内容中对“数组”的阐述，或暂时跳过这一节。

如上文所述，界面上的元素通常有嵌套的组合关系。一个大的界面元素中，还可能包含了多个小的界面元素，我们称之为“子元素”。同样，子元素当中，也可能又包含了多个子元素，不妨称之为“孙元素”。

我们回到之前用来做实验的“员工信息表”。可以想象，在这个界面上，整个表格就是一个界面元素，其中的每一行都是一个子元素，而各行的每个单元格又是一个孙元素。实际情况会比我们想象的稍微复杂一些，因为这个表格的层级关系会更多一些，比如每一行可能已经是整个表格的孙元素了，而每个单元格又是孙元素的孙元素了。

可以实际测试一下，通过鼠标拖动，将“获取子元素”命令加入到可视化视图中，并仔细移动鼠标，把目标元素设为整个表格，如下图所示：

ID	NAME	SALARY	EMAIL
1	朱元璋	8100	zhuyz@ming.com
2	赵匡胤	8800	zhaoky@song.com
3	李世民	5500	lism@tang.com
4	刘邦	7000	liub@han.com
5	嬴政	8500	yingz@qin.com

图 43: 把目标元素设为整个表格

此时，在这条命令的右侧，有一个蓝色的三角形（见下图中的红色框）。点击这个三角形，可以仅运行当前这一行。这里有一个小小的技巧：在运行某一行命令的时候，如果这一行命令有“输出到”属性，UiBot还会自动把输出的内容显示在下面的“输出”窗口上，以便调试。

在这条命令的属性中，有一个非常重要的属性叫“子元素层级”（见下图中的黄色框），这个属性通常是一个整数。当这个属性为1的时候，代表“获得所有子元素”；当这个属性为2的时候，代表“获得所有孙元素”；当这个属性为3的时候，代表“获得所有曾孙元素”；以此类推。当然，这个属性还可以为0，代表“获得所有子元素、孙元素、曾孙元素等等，直到没有下一级元素为止”。

我们可以试试，当“子元素层级”属性为2的时候，可以得到表格中所有的行，当“子元素层级”属性为4的时候，可以得到表格中所有的单元格。这条命令的输出值实际上是一个数组，数组的每个元素代表一个界面元素，还可以把数组中的每个元素直接用在其他的“界面元素操作”命令中。比如，我们要让鼠标依次双击这个表格中的每个单元格，可以使用下面的流程来实现（请留意右侧的“目标”属性）：

可视化 源代码

- 获取目标元素下子元素层级为 4 级范围内的所有元素, 输出到 `arrElement` 表格<table>_ID
- 用 `value` 遍历数组 `arrElement`
- 鼠标点击目标 **重新指定**
- 等待 500 毫秒后继续运行

目标 ②
Exp value x

鼠标点击 ②
Exp 左键

点击类型 ②
Exp 双击

图 44: 点击表格中的每个单元格

界面元素的操作还包括设置/获取元素选择、设置/获取元素属性、设置/获取元素文本、获取元素区域、元素截图等等。其使用方法可查阅UiBot命令手册，在这里不再赘述。

3.5 UI分析器

在前文中提到，界面上的元素通常有嵌套的组合关系。一个界面元素中，还可能包含了多个“子元素”，而“子元素”又包含了“孙元素”。

对于比较简单的界面，一般直接选择界面元素就够了。但有的界面特别复杂，比如下图所示的是某个电商网站的商品界面，其中包含了许许多多的界面元素，有图片，有图标，有各种各样的文字，甚至还有浮在图片上面的文字！这些界面元素的特征各不相同，嵌套关系也错综复杂，稍不注意，就容易搞错。



图 45: 某购物网站商品列表

为了避免“错选”和“漏选”，一个有效的办法是仔细分析界面元素的嵌套关系，看看我们需要作为目标的界面元素都有哪些子元素，哪些“兄弟元素”，它的“父元素”又是谁。比如下面是两条常用的技巧：

- “错选”一般都发生在“兄弟元素”之间，本来要找某个界面元素，结果找到的是它的“兄弟”。如果我们仔细观察界面元素和它的“兄弟”的特征有何不同，就容易找到区分它们的特征，避免“错选”。
- “漏选”一般都是因为在界面元素的特征中，某个属性会经常变化。有的时候，虽然我们要找的界面元素的属性会经常变化，但它的父元素或子元素的属性却不容易变化。我们可以先把相对比较稳定的父元素或子元素作为目标来查找，然后再采用“获取子元素”或“获取父元素”的命令，来找到我们真正需要的界面元素（当然，“获取子元素”得到的是一个数组，其中可能有不止一个元素，还要考虑进一步区分）。以避免“漏选”。

使用“UI分析器”，可以方便地查询界面元素及其特征，还能轻松地定位到父元素、子元素或兄弟元素。UI分析器是一个独立的应用程序，在UiBot Creator中，有好几处入口都可以用来启动UI分析器。这些入口包括：

- 在UiBot Creator的首页中，点击“工具”标签，可以找到UI分析器的“启动”按钮。
- 在编写任意一个流程时，在工具栏上，可以找到“UI分析器”的按钮。
- 在编辑任何一个界面元素的属性时，可以找到“从UI分析器打开”的按钮。

采用前两种方式打开UI分析器时，UI分析器中暂时未选中任何界面元素。采用第三种方式打开时，则会自动选中正在编辑的界面元素。除了上述差异之外，以上几种方式打开的UI分析器都是一模一样的。UI分析器的界面大致是下图所示的样子：

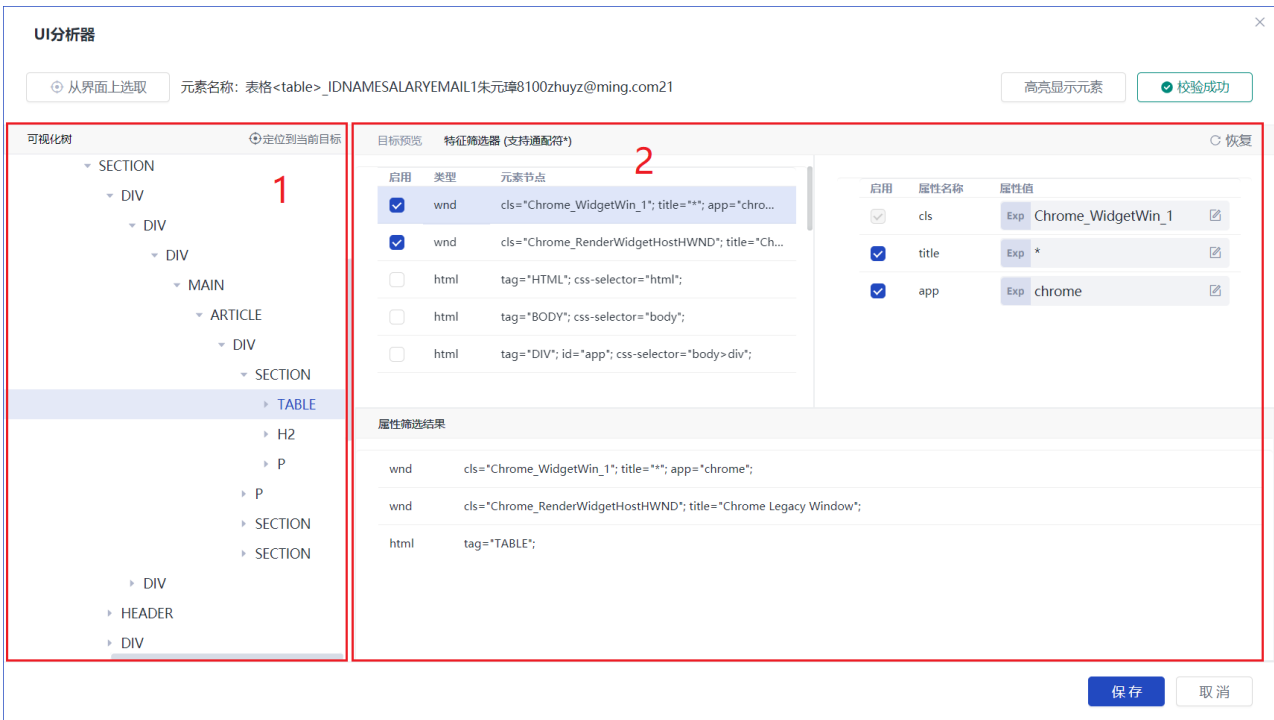


图 46: UI分析器的界面

看起来有些复杂，但实际上如果您阅读过前文中关于编辑界面元素的内容，对于上图中的区域2应该已经不陌生了。稍有不同的是：在上图的区域2中，会把所有可能用到的界面元素的属性都列出来，比编辑界面元素时显示得更多，以便您仔细挑选合适的属性组合，避免“错选”和“漏选”。

在UI分析器中，更重要的是上图中的区域1。这个区域是一个树形结构，称为“可视化树”，树中的每个节点代表了一个界面元素，各个节点在树中的父子关系，就代表了它们在界面上的父子关系。可以通过点击节点左边的小三角形展开下面的子节点。

我们在使用UI分析器的时候，通常可以进行以下操作：

- 在可视化树中，查看节点之间的父子、兄弟关系。
- 右键单击可视化树中的某个节点，并选择“设置为目标元素”，使这个界面元素的属性显示在区域2之中。

- 修改界面元素的属性之后，使用“校验目标”的功能，验证您的修改是否仍然能找到界面元素。
- 通过“高亮显示元素”的功能，显示界面元素的位置。
- 如果当前已经打开了某个流程，则可以使用“保存到界面库”的功能，将区域2里面的界面元素保存到该流程的界面库中。

对于复杂的界面，通常需要组合以上操作，仔细分析，认真思考。也需要经常练习，积累经验，才能在实战中游刃有余。作为思考题，请读者再次考虑：如果要找到“员工信息表”中“李世民”的薪资，并进行双击。借助UI分析器，使用尽可能多种方式来实现！

3.6 安装扩展

在用遮罩选取目标的时候，很常见的一种情况是：遮罩无论如何只能遮住整个窗口，或窗口的客户区，而无法选取里面的具体界面元素。如下图，只能选中Chrome浏览器的整个页面，不能选取里面的输入框和按钮。

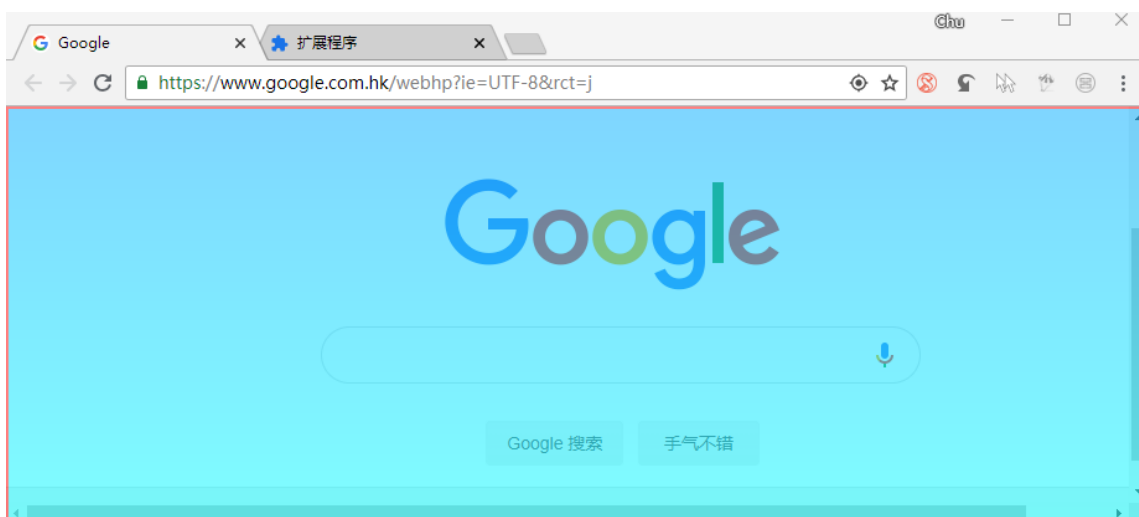


图 47: 用Chrome浏览器，无法选取具体目标

对于这种情况，有两种可能性，一种是界面真的无法选取（我们会在下一章详细讨论），另一种是界面其实可以选取，但是还需要做一些附加的工作，比如对要操作的软件安装扩展程序之后，才可以选取之前无法选取的界面元素。目前，这些扩展程序包括Chrome浏览器扩展，Firefox浏览器扩展和Java程序扩展。安装了这些扩展以后，就可以选取Chrome浏览器中的页面元素，Firefox浏览器中的页面元素和Swing、AWT、JNPL、Applet等多种Java应用程序界面元素了。

下面我们以Chrome浏览器为例，看看这些扩展程序的安装方法。

3.6.1 Chrome浏览器

Chrome浏览器需要安装扩展程序，并启用了扩展程序，才能正常选取。请留意您的Chrome浏览器地址栏右边的一排小图标，要有如下图所示的这个图标（颜色可能是灰色，但不影响正常工作）才行，鼠标

移动上去，还有文字提示“UiBot Native Message Plugin”。



图 48: UiBot在Chrome浏览器上的扩展图标

如果没有安装扩展程序，则需要遵循以下步骤进行安装：

1. 关闭Chrome浏览器；
2. 打开UiBot Creator，随便选择一个流程。在菜单中选择“帮助”->“安装扩展”，选择“Chrome扩展”；
3. 打开Chrome浏览器，稍等片刻，浏览器会提示已添加新的扩展，如下图。此时请务必选择“启用扩展程序”；

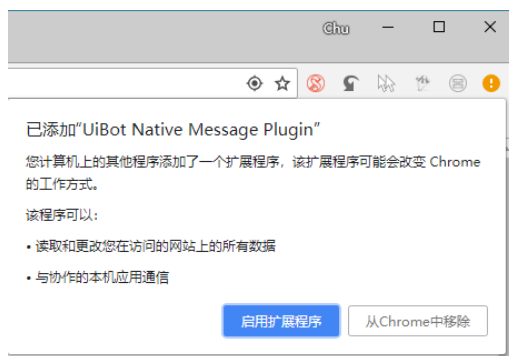


图 49: Chrome浏览器提示添加新的扩展

4. 如果仍然有问题，请按照下图所示，打开Chrome的扩展程序管理功能，并启用UiBot Native Message Plugin。

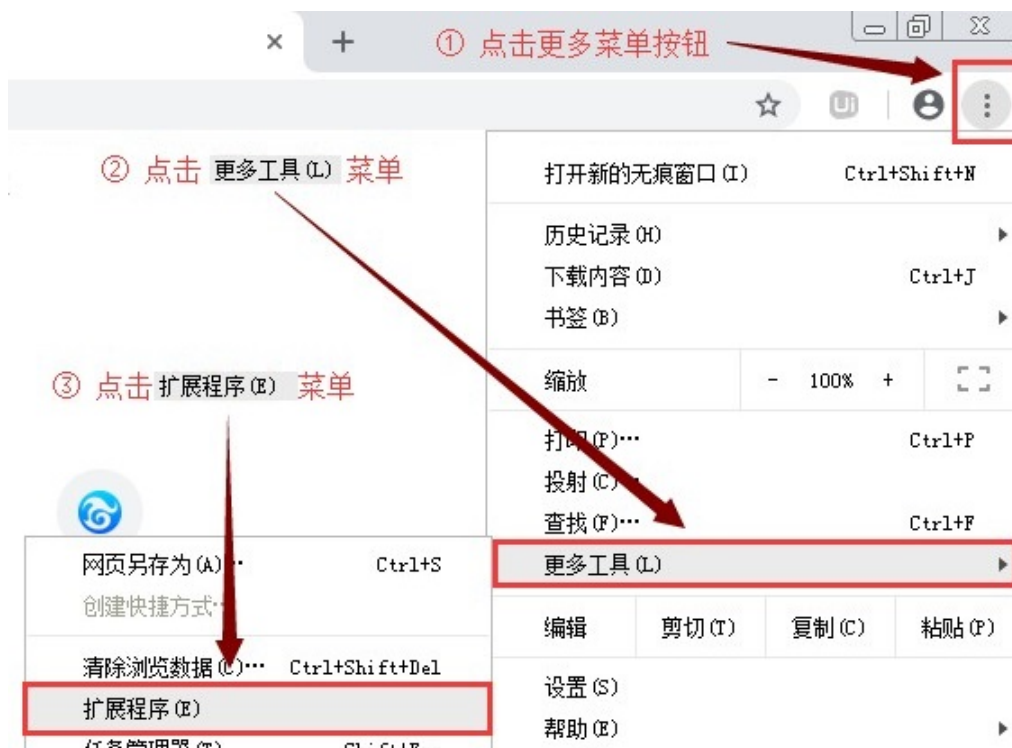


图 50: Chrome浏览器的扩展管理

除了IE、Chrome浏览器之外，我们还经常用到百度浏览器、360安全浏览器、QQ浏览器等国产浏览器。这些浏览器都采用了Chrome内核或IE内核，理论上UiBot也可以支持获取其中的界面元素。但是，由于其设置方式各不相同，还经常发生变化，为了简单起见，推荐在RPA流程中使用原生的Chrome浏览器或IE浏览器。

3.6.2 SAP程序

UiBot支持SAP产品的录制及自动化操作，为了能够识别、操作SAP控件需要在开始之前对SAP进行一些设置。

3.6.2.1 开启SAP GUI Scripting (服务端)

1、登录SAP，执行事务RZ11

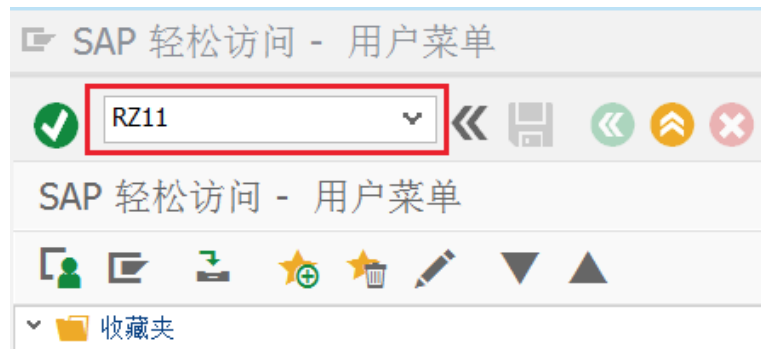


图 51: 执行事务RZ11

2、输入参数sapgui/user_scripting，并点击“显示”

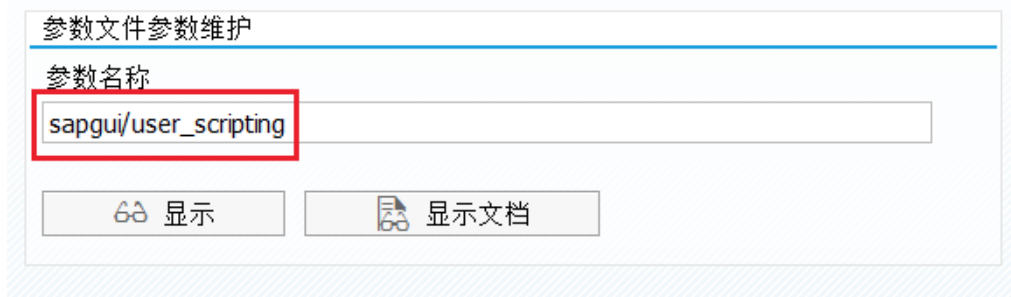


图 52: 输入参数sapgui/user_scripting

3、点击“更改值”，在“新值”中输入TRUE

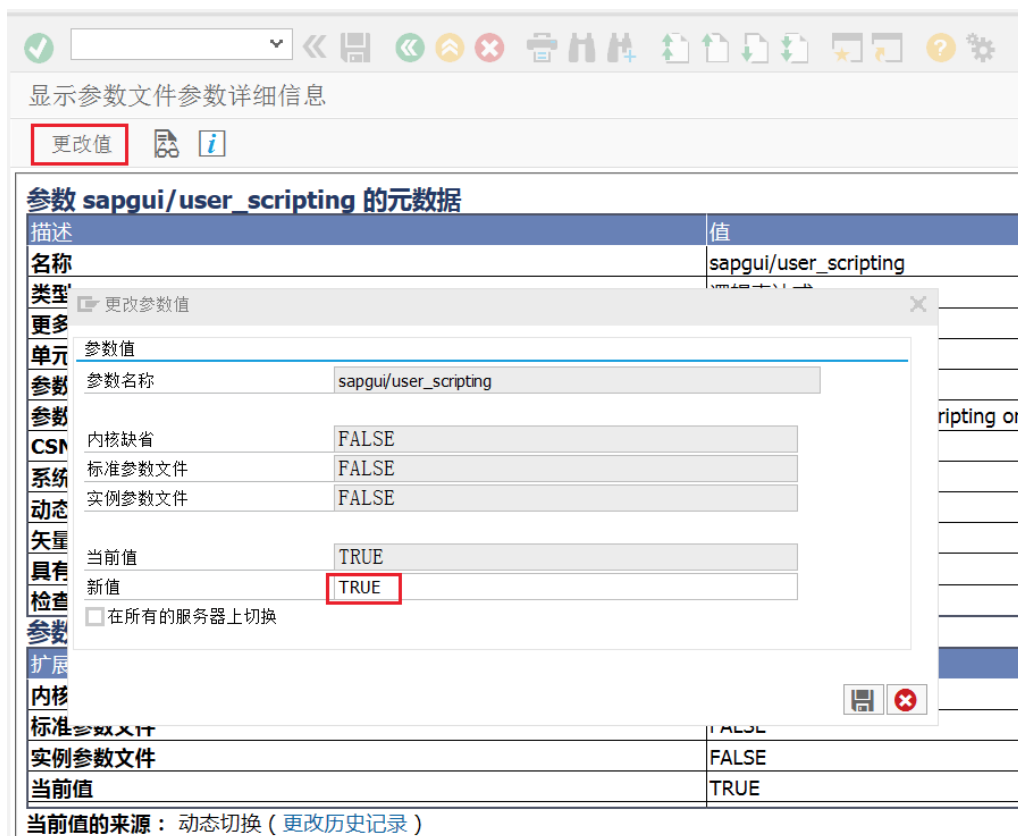


图 53: 更改值

3.6.2.2 开启SAP GUI Scripting (客户端)

- 1、登录SAP，点击“定制本地布局(Alt+F12)”，选择“选项”



图 54: 定制本地布局

- 2、切换到“辅助功能与脚本”—“脚本”选项，在“用户设置”中选中“启动脚本”，并屏蔽其他通知

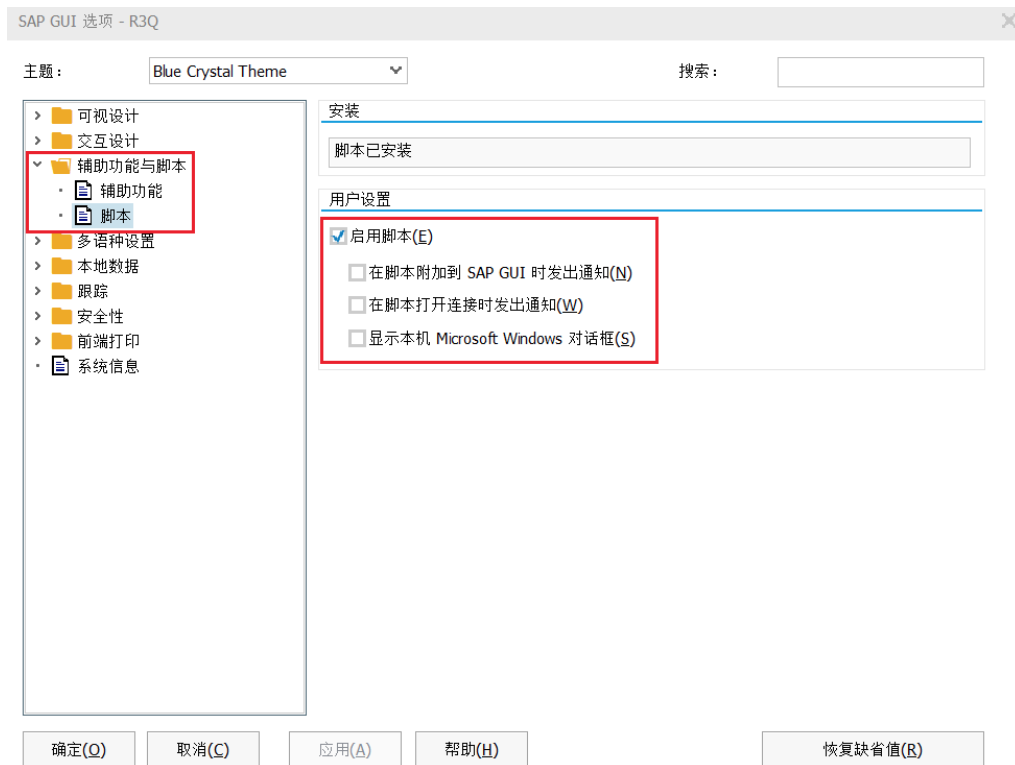


图 55: 辅助功能与脚本

3.6.2.3 F4帮助设置为模态对话框

- 1、登录SAP，点击“帮助”—“设置...”菜单
- 2、切换到“F4 帮助”，在“显示”设置中选“对话 (模式)”



图 56: F4帮助设置

4 界面图像自动化

在上一章中，我们讲述了“界面元素”，以及如何选取一个界面元素作为目标，以便使用“界面元素自动化”。当然，并非在所有的情况下，都能准确找到恰当的界面元素作为目标。因此，我们需要学会使用“界面图像自动化”，以备不时之需。

4.1 为什么不能使用界面元素？

在上一章中提到，我们在查找、操作界面元素的时候，实际上都是在调用界面元素所在的软件给我们提供的接口。UiBot所做的，实际上是把这些不同种类的接口统一起来，让编写流程的人不需要关注这些细节。但是，仍然会有一些软件，没有给我们提供查找、操作界面元素的接口；或者虽然提供了接口，但在最终发布时关闭了，这些软件包括：

- 虚拟机和远程桌面

包含Citrix、VMWare、Hyper-V、VirtualBox、远程桌面（RDP）、各种安卓模拟器（如腾讯安卓模拟器）等。这些程序都由单独的操作系统在运行，和UiBot所在的操作系统是完全隔离的，UiBot自然无法操作另一个操作系统里面的界面元素。

当然，如果条件允许的话，可以把UiBot和流程涉及到的软件，都安装在虚拟机里，或者远程计算机里。这样一来，这些软件提供的接口就可以被UiBot直接使用了，因为它们还是运行在同一个操作系统里面的，本地计算机只是起到了一个显示器的作用而已。

- 基于DirectUI的软件

以前，Windows软件界面的开发框架都是微软提供的，包括MFC、WTL、WinForm、WPF等。微软很贴心的为这些框架制作出来的界面都提供了自动化操作的接口。近年来，为了让软件界面更好看，也更容易制作，很多厂商或开发团队推出了自己的Windows软件界面开发框架。这类框架统称为DirectUI。用这些框架制作的界面，其界面元素都是“画”出来的，虽然人眼可以看到，但操作系统和其他程序都不知道界面元素到底在哪里。有的DirectUI框架提供了对外的接口，可以找到界面元素，有的则根本没有提供这样的接口，其它程序，包括UiBot，自然也无法找到界面元素。

实际上，UiBot Creator、UiBot Worker本身的界面就是用一种DirectUI框架开发的，这种框架称为Electron。Electron其实提供了界面元素的查找接口，但对外发布的版本默认都关闭了。所以，细心的读者可能会发现，UiBot里面的界面元素，反而是市面上任何RPA平台都无法找到的。

- 游戏

由于游戏的界面强调美观和个性化，所以，一般游戏的界面元素都是“画”出来的，原理上和DirectUI类似。这种界面通常也没有提供接口，告知我们界面元素的位置。和基于DirectUI的软件不同的是，游戏界面变化速度快，对时效性的要求更高，一般来说，RPA平台并未针对游戏进行优化，所以在游戏上使用不会太好。

如果要在游戏上使用自动操作，推荐使用按键精灵。按键精灵是专门为游戏设计的，内置了很多针对游戏的界面查找手段，比如单点颜色比对、多点颜色比对、图像查找等等。且运行效率更高。

4.2 无目标命令

我们在上一章中介绍了“有目标”的命令，相对的，UiBot也有“无目标”的命令。如下图所示，红框中表示有目标的命令，蓝框中表示无目标的命令。

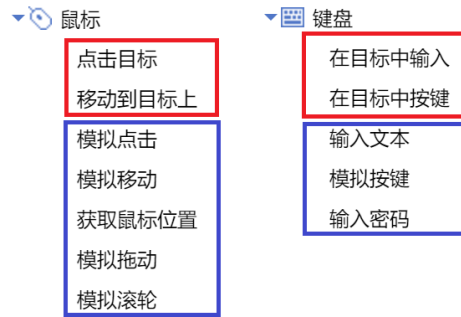


图 57: 有目标和无目标的命令

如果遇到了没有界面元素作为目标的Windows软件，“有目标命令”自然就不能再用了，但仍然可以用“无目标命令”。在图中这些无目标的命令里面，最重要的是“模拟移动”，因为“模拟移动”需要在命令中指定一个坐标点，在执行这条命令的时候，鼠标指针也会移动到这个坐标点；移动之后，我们再使用“模拟点击”命令，模拟按下左键，才能正确的按下某个按钮；或者正确的在某个输入框上设置焦点，之后，再使用“输入文本”命令，才能在焦点所在的输入框里面输入一段文本。

比如，有一个输入框，其中间的坐标是x:200, y:300。那么我们就需要先用“模拟移动”，并设定移动的坐标为x:200, y:300；再用“模拟点击”按下左键，设置焦点；再用“输入文本”，才能正常输入。否则，直接用“输入文本”的话，很大概率就输入到其他输入框里面了。

这里，我们有必要先解释一下Windows操作系统的屏幕坐标系。如果您之前了解Windows的屏幕坐标系，这一段可以跳过不看。

在Windows操作系统中，屏幕上的每一点都有一个唯一的坐标，坐标由两个整数组成，一个称为x，另一个称为y。例如坐标x:200, y:300的含义就是这个点的坐标的x值是200，y值是300。x是以屏幕左边为0开始计算，从左到右分别是0,1,2,3...，以此类推。y是以屏幕上边为0开始计算，从上到下分别是0,1,2,3...，以此类推。所以，坐标x:200, y:300所对应的点，其位置大致如下图中红圈所示：

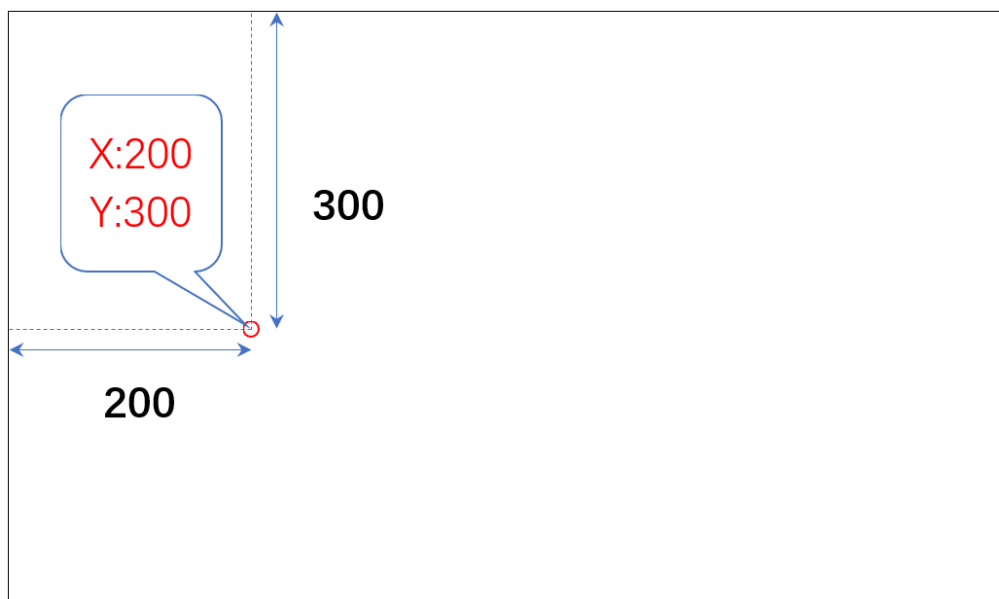


图 58: Windows的屏幕坐标系

只要有x和y两个整数值，就可以确定屏幕上一个点的位置。在UiBot中，有一些命令可以获得屏幕上某点的位置，并输出到一个变量里。如何用一个变量来保存x和y两个值呢？我们在后面学习UiBot所使用的BotScript语言的时候会了解到，BotScript中有“字典”数据类型，可以保存多个值。所以，UiBot在输出一个点的位置的时候，会输出到一个字典类型的变量中。假设这个变量名为pnt，则使用pnt["x"]和pnt["y"]即可得到坐标的x和y两个值。

假如我们要找的界面元素在屏幕上的固定位置，那么用固定的坐标，配合无目标命令，即可正常模拟操作。但这种情况往往比较少见，因为Windows是多窗口系统，每个窗口的位置都可以被拖动，导致窗口里面的界面元素的位置也会发生变化。而且，在微信这样的软件中，联系人的位置也不是固定的，而是根据最近联系的时间排序的，位置随时可能发生变化。

所以，在UiBot中，一般不推荐直接写固定的坐标，因为变化的情况太多了，很难一一考虑周全。通常，如果使用无目标的命令，需要搭配其他命令使用，让其他命令能根据某种特征，找到界面元素的坐标，然后把找到的坐标当作变量，传给这些无目标命令。

在UiBot中，无目标命令的最佳拍档，是图像命令。两者结合使用，才能实现“界面图像自动化”。

4.3 图像命令

除了常用的“鼠标”、“键盘”类之外，UiBot的“图像”类命令也是很强大的。在UiBot Creator的命令区，找到“图像”，单击展开，可以看到其中包含了如下图所示的几个命令：



图 59: UiBot Creator里面列出的“图像”类命令

我们首先来看“查找图像”这条命令，其作用是：首先指定一个图像文件，格式可以是bmp、png、jpg等（推荐使用png格式，因为它是无损压缩的），然后在屏幕上的指定区域，按照从左到右，从上到下的顺序依次扫描，看这个图像是否出现在指定区域当中。如果出现，则将其坐标值保存在一个变量中，否则发生异常。

看起来好像很复杂，又要指定图像文件，又要指定扫描的区域。实际上，使用UiBot Creator的话，操作非常简单。

比如，著名的游戏平台Steam，其界面就采用了DirectUI技术。我们以其登录对话框为例（如下图），其中的账户输入框、密码输入框、登录按钮等元素都无法被任何RPA工具直接获取到。这时候就需要用到图像命令。



图 60: Steam的登录对话框

假设已经启动了Steam，并打开了其登录界面，且Steam已经自动保存了有效的用户名和密码，只差点点击“登录”按钮了。下面，在UiBot Creator中编辑一个流程块，并以双击或拖动的方式，插入一条“查找图像”命令，点击命令上的“未指定”按钮并在弹出菜单中选择“从界面上选取”：

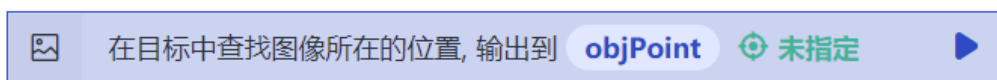


图 61: 使用“查找图像”命令

和有目标命令类似，UiBot Creator也会暂时隐藏，图标会变成一个箭头和一张图片的样子。此时，按下左键，并向右下方拖动，直到画出一个蓝框，且蓝框中已经包含了要找的图像，松开鼠标左键，大功告成！



图 62: “查找图像”命令指定图像和查找范围

看起来，上面的操作只是画了一个蓝框，但是，UiBot Creator已经帮我们做了两件事情：

1. 判断蓝框落在哪个窗口上，并记录这个窗口的特征，将来找图的时候，也需要先找到这个窗口，并在这个窗口的范围内找图。
2. 对蓝框所框住的部分截图，自动保存为一个png格式的文件，并自动把这个文件保存在当前所编写的流程所在目录的res子目录中。这就是将来要查找的图片。

用鼠标单击这条“查找图像”命令，将其置为高亮状态，右边的属性栏会显示出这条命令的属性，如下图所示：

图 63: “查找图像”命令的属性

其中，画红框的两条属性，也是最重要的两条属性，就是前面所说的，UiBot Creator帮我们做的两件事情。其他各个属性里面，“相似度”是一个0-1之间的数字，可以包含小数位，这个数字越接近1，UiBot在查找图像时，越严格要求每个点都必须匹配上，通常取0.9，表示允许出现一小部分不匹配的情况，只要大体匹配即可。“光标位置”属性的含义是，当找到图像时，由于图像是一个矩形，而命令输出只是一个点的坐标，究竟要返回矩形中的哪个点的坐标，通常取“中心”即可。“激活窗口”属性表示在找图之前，是否需要先把所查找的窗口放到前台显示。如果窗口被遮住了，即使窗口上有我们要找的图像，也无法正确找到，所以这个属性通常也设为“是”。

其他的属性通常不用改，保持默认值就好。在“输出到”属性中，已经指定了一个变量名objPoint，如果成功的找到了图像，会把结果保存在这个变量中。我们来看看这个变量中保存了什么内容：在命令区的“基本命令”类中找到“输出调试信息”，将其插入到查找图像命令的后面，并且在属性中指定输出内容为objPoint（注意，此时应将属性左边的标有“Exp”的按钮切换为蓝色，表示这是可以输入变量和表达式的“专业模式”，然后输入变量名objPoint，或者按“fx”按钮直接选择变量）。如图所示：



图 64: 用“输出调试信息”查看结果

假设要查找的图像确实能在屏幕上看到，运行这个流程块后，得到结果：

```
{ "x" : 116, "y" : 235 }
```

具体的数值在不同的计算机上可能有所不同，但原理不变。这个值是一个“字典”数据类型，当这个值保存在变量objPoint中的时候，只需要写 `objPoint["x"]` 和 `objPoint["y"]` 即可得到其中的x和y值。

下面，得到了图像的中心位置，只需要用鼠标去点击这个位置，即可模拟Steam的登录操作了。选用“鼠标”类中包含的“模拟移动”和“模拟点击”命令，即可很好的完成任务。

如下图，“模拟移动”命令最关键的属性，就是要操作的屏幕位置。将“横坐标”和“纵坐标”两个属性分别切换到“专业模式”，并依次输入查找图像的结果 `objPoint["x"]` 和 `objPoint["y"]` 即可。移动完成后，再来一个“模拟点击”，让左键在登录按钮的中心点下去。至此，我们已经模拟出点击“登录”按钮的全套操作。



图 65: 一套完整的“查找图像并点击”操作

上面三条命令很容易看懂，即使是从来没有学过UiBot的用户，也能大致了解其含义。但是，仅仅为了点一个登录按钮，还需要三条命令才能完成，显然过于复杂了。这时候，请再回头看一下UiBot提供的“图像”类下的所有命令，其中第一条命令叫“点击图像”，它其实就是“查找图像”、“模拟移动”、“模拟点击”三条命令的组合，只要插入一条“点击图像”命令，并按下命令上面的“从界面上选取”按钮，拖动鼠标选择要查找的窗口和要查找的图像，即可快速完成模拟点击Steam的“登录”按钮的功能。虽然是无目标的命令，但其操作便捷程度并不逊于有目标的命令。

有了上述基础，对于其他几条图像类的命令，包括“鼠标移动到图像上”、“判断图像是否存在”等等，您应该可以举一反三了，本文不再赘述。

4.4 实用技巧

在上一章中，我们学习了界面元素自动化，而在这一章中学习了界面图像自动化。我们看到，在大多数

情况下，并不能单纯的使用“无目标命令”，而是要结合图像类命令，动态的在屏幕上找到要操作的位置，所以才称之为“界面图像自动化”。

那么，在具体完成一个流程任务的时候，该优先选择界面元素自动化，还是优先界面图像自动化呢？我们给出的答案是：优选界面元素自动化！只要能获得恰当的界面元素作为目标，就应该优先考虑使用界面元素。因为使用界面图像自动化，有以下的缺点：

- 速度通常会慢于界面元素自动化；
- 可能受到遮挡的影响，当图像被遮挡时，即使只遮挡了一部分，也可能受到很大影响；
- 往往需要依赖图像文件，一旦丢失图像文件就不能正常运行；
- 某些特殊的图像类命令必须连接互联网才能运行。

当然，这些缺点也是可以部分缓解的，以下技巧能帮您更好的使用图像类命令：

首先，请牢记一个“小”字。在截图时，尽量截取较小的图像，只要能表达出所操作的界面元素的基本特征即可。在指定查找的区域时，尽量缩小区域。这样不仅速度会有所改善，而且也不容易受到遮挡的影响。比如下图中的“登录”按钮，没必要像左图一样，把整个按钮作为一幅图像来查找，只要像右图一样选择最关键的部分就可以了。



图 66: 选择较小的截图

其次，大部分图像命令都支持“相似度”的属性，这个属性的初始值是0.9，如果设置过低，可能造成“错选”，如果设置过高，可能造成“漏选”（“错选”和“漏选”的概念请参考上一章）。可以根据实际情况进行调整，并测试其效果，选择最佳的相似度。

再次，屏幕的分辨率和屏幕的缩放比例对图像命令可能有非常关键的影响。因为在不同的分辨率下，软件的界面显示可能完全不一样，导致图像命令失效。所以，请尽量保持运行流程的计算机和开发流程的计算机的分辨率、缩放比例都是一致的。在Windows 10操作系统上设置分辨率和缩放比例的界面如下图所示：

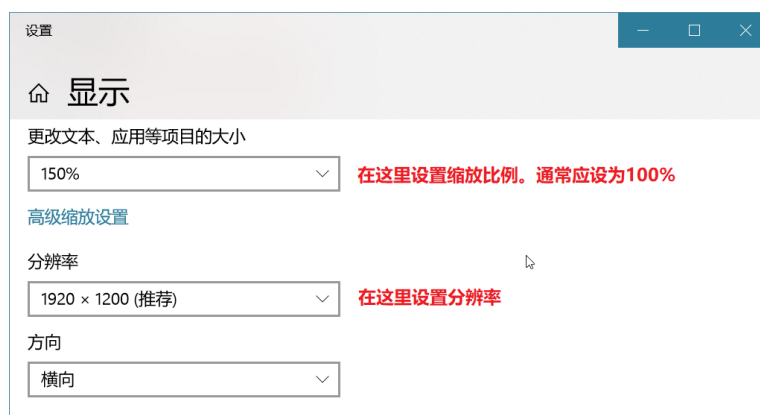


图 67: 设置分辨率和缩放比例

最后，对于图像命令来说，经常需要和图像文件打交道。当需要使用图像文件时，我们固然可以用一个绝对路径来测试，如D:\1.png。但是，这就要求在运行此流程的计算机上，也必须在同一路径下有同样的文件，否则就会出错。有一个改进的方法，就是在您的流程所在的文件夹中，可以看到一个名为res的文件夹，把图像或其他文件放在这个文件夹中，并在流程中使用表达式@res"1.png"来代表这个文件即可。这样的话，当前流程发布到UiBot Worker使用的时候，也会自动带上这个文件。并且无论UiBot Worker把这个流程放在哪个路径下，都会自动修改@res前缀所代表的路径，使其始终有效。

另外需要说明的是，本章所描述的图像类命令使用技巧，绝大部分也适用于OCR命令，关于OCR命令的概念和使用方法，因篇幅所限，本章不再赘述。

4.5 智能识别

如前所述，虚拟机、远程桌面、基于DirectUI的软件、游戏等应用程序，无法直接使用有目标命令的“从界面上选取”功能定位界面元素。在这种情况下，只能使用无目标命令和图像命令配合的方式，但图像命令有一些使用技巧不易掌握，在掌握不好的情况下，非常容易出现“错选”或者“漏选”。因此，UiBot Creator从5.0版本开始，支持智能识别功能，这是另一种基于图像进行界面元素定位的方法。我们先从一个具体实例来看看智能识别的用法。

打开Windows自带的画图程序，绘制一个矩形框，假设此时的需求是：通过UiBot找到并点击这个矩形框。

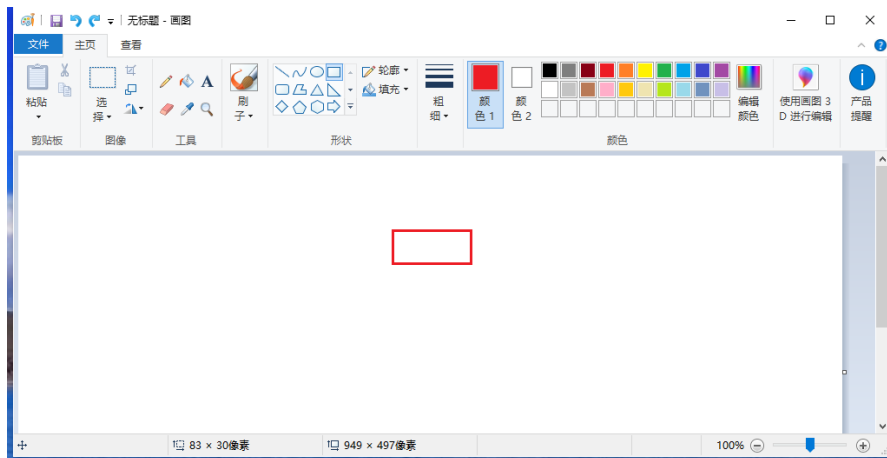


图 68: 一个按钮的界面

如前所述，有目标命令是无法找到这个矩形框的，我们来看看如何通过智能识别命令找到并点击这个矩形框。在UiBot Creator的命令区，找到“界面操作”，单击展开，找到“智能识别”，再单击展开，可以看到其中包含了如下图所示的一组命令：

- ▼ 智能识别
 - ◇ 智能识别屏幕范围
 - ◇ 智能识别后点击
 - ◇ 智能识别后获取文本
 - ◇ 智能识别后输入文本
 - ◇ 智能识别后鼠标悬停
 - ◇ 智能识别后判断元素存在

图 69: 智能识别命令列表

首先插入一条“智能识别屏幕范围”命令，然后点击这条命令上的“查找目标”按钮，UiBot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，鼠标移动到什么地方，这个目标选择器就出现在什么地方。细心的同学已经发现这个功能跟有目标命令的“从界面上选取”按钮的功能是一样的！

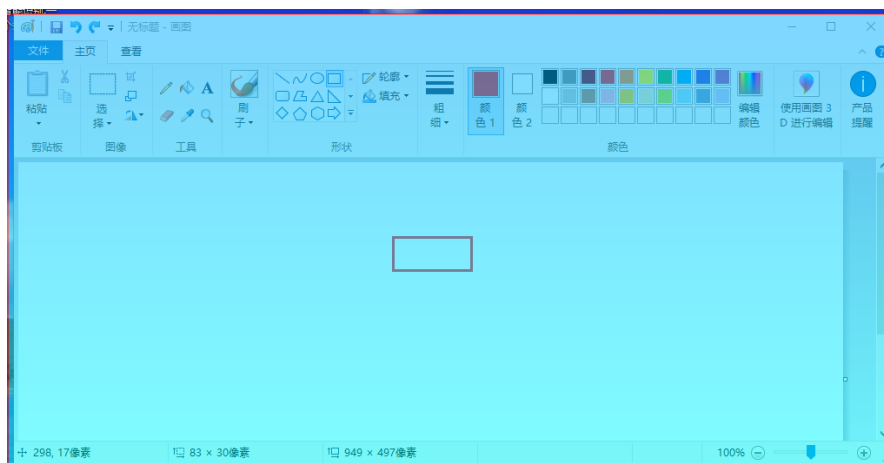


图 70: 智能识别屏幕范围

选取完屏幕后，再插入一条“智能识别后点击”命令，然后点击这条命令上的“查找目标”按钮，这里的“查找目标”按钮的用法仍然与有目标命令的“从界面上选取”按钮相同。这个时候神奇的现象出现了：UiBot居然将刚才我们绘制的一个矩形框认出来了！

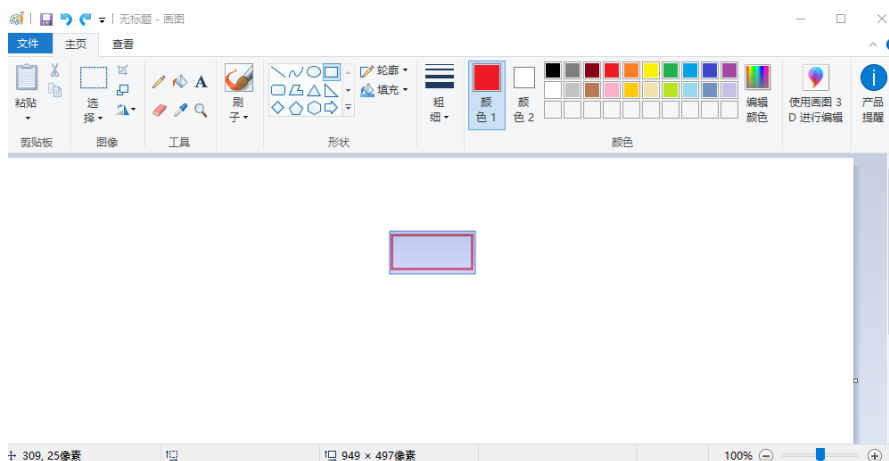


图 71: 智能识别后可以找到矩形框

也就是说，通过“智能识别屏幕范围”命令，UiBot将原来无法识别的界面，通过人工智能图像识别，将界面中一个个潜在的元素给提取出来，并供后续的命令使用，这些后续命令包括“智能识别后点击”、“智能识别后获取文本”、“智能识别后输入文本”、“智能识别后鼠标悬停”、“智能识别后判断元素是否存在”等命令。从这个角度也可以理解，“智能识别后点击”等命令，必须在“智能识别屏幕范围”命令之后执行，且必须在“智能识别屏幕范围”命令的范围内才有效（在“智能识别屏幕范围”命令缩进范围内）。



图 72: 智能识别多条命令组合使用

运行该流程，可以看到成功地点击了该矩形框。

如果用户界面中存在两个或两个以上外观相同的界面元素，UiBot如何定位我们想要找的那个界面元素呢？我们还是通过具体实例来讲解：打开画图程序，把刚才绘制的矩形框再复制一份，这样画图界面中就同时存在两个一模一样的矩形框了，假设现在的需求是：通过UiBot找到并点击右边那个矩形框。

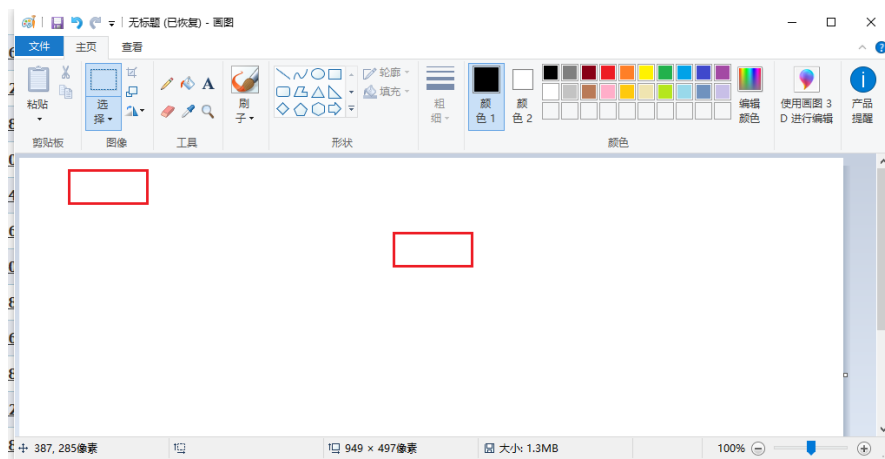


图 73: 两个按钮的界面

这时，需要重新点击“智能识别屏幕范围”命令的“查找目标”按钮，因为需要识别的屏幕内容已经发生变化，需要对屏幕重新进行智能识别。从这个角度来看，“智能识别屏幕范围”命令其实是一个预先执行的静态命令，而不是流程运行过程中动态进行查找目标。所以一旦屏幕图像有变化，都需要对屏幕图像重新进行智能识别。

然后，再点击“智能识别后点击”命令的“查找目标”按钮。这个时候我们可以发现，两个矩形框都处于可以选择的状态。我们选择右边那个矩形框，此时右边矩形框被遮罩框遮住，同时有一条虚线，将右边矩形框与一个“形状”字样连接起来，如下图所示：

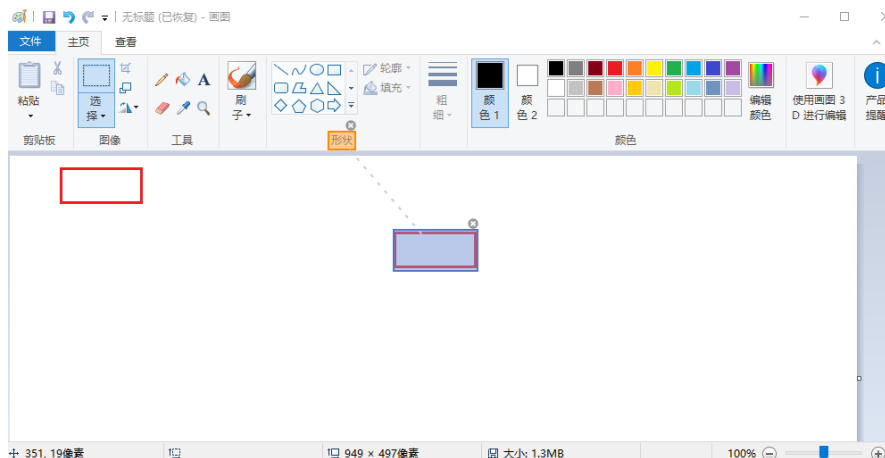


图 74: 在两个相同矩形框中定位某个矩形框

原来，UiBot使用了一种叫做“锚点”的技术，来定位两个或多个外观相同元素的位置。所谓“锚点”，指的是屏幕中某个独一无二的元素（比如上述“形状”字样），利用不同矩形框相对于该锚点的位置偏移和方位角的不同，即可唯一地定位矩形框。

运行该流程，可以看到成功地点击了右边的矩形框。

5 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，学习了界面元素自动化这一章，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，UiBot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，只需要一两条命令就可以做到。

用UiBot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。

本章假设读者对浏览器、Word、Excel、数据库等软件及相关知识已经有初步的了解，最好是在工作中使用过这些软件。如果还缺乏了解，市面上有大量书籍可以参考，本文不另行介绍。

5.1 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文件对应一个工作簿，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，就是说工作簿和工作表是一对多的关系。

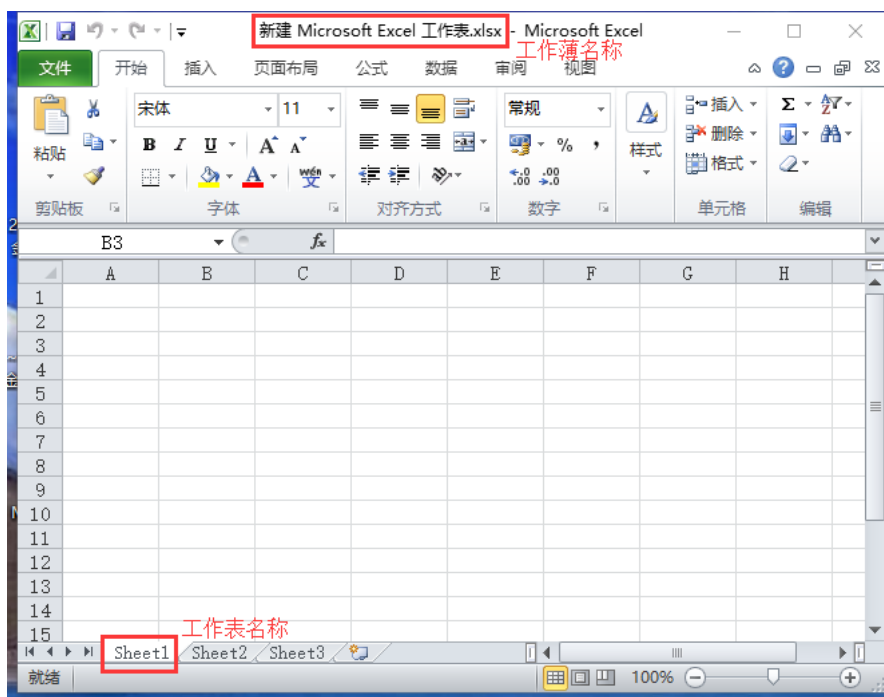


图 75: Excel工作簿和工作表

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。

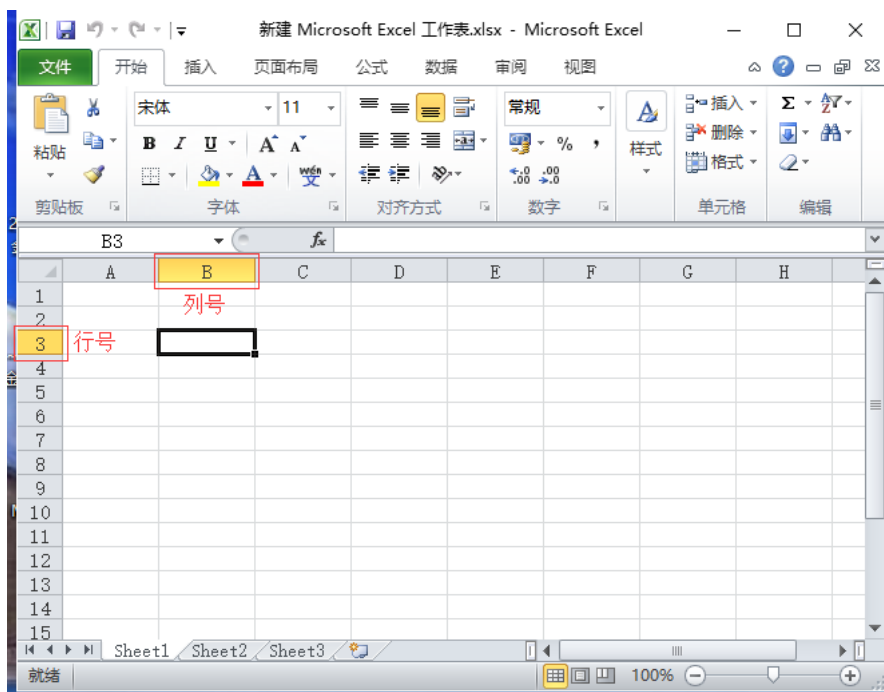


图 76: Excel的行和列

用UiBot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UiBot打开一个工作簿。在UiBot Creator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，也可以切换到专业模式，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。请注意，如果使用绝对路径，推荐切换到普通模式，并点击右侧的文件夹图标按钮，直接选取文件即可，比较简便且不容易出错。否则，如果切换到专业模式，就需要按照字符串的格式来写，不仅要用引号表示这是一个字符串，还需要把路径中的\符号写为\\。



图 77: 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。

另外两条属性是指定要打开的Excel工作簿的密码，如果没有密码，保存空白即可。

在上面的“输出到”属性中，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，可以看到这条命令的属性如下图所示：



图 78: 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。

另外，我们需要指定“工作表”和“单元格”属性，来告诉UiBot要读取哪个工作表中的哪个单元格。“工作表”和“单元格”的指定方式也分两种：一种是按照Excel的习惯，用"Sheet1"和"A1"，分别指代名为“Sheet1”的工作表，以及名为“A1”的单元格。

还有第二种方式，就是用一个整数来指代工作表。UiBot中通常是从0开始编号的，所以整数0代表的是第一张工作表，整数1代表的是第二张工作表，以此类推，而不管这个工作表实际上叫什么名字。也可

以用中括号里面的两个整数，例如[x, y]这种方式（实际上是一个数组，如果不了解数组的概念，可以暂时不用关心这个细节），来指代单元格。其中x和y都是以1开始编号（这里以1开始编号，是为了适应Excel的习惯）的整数，x代表的是行，y代表的是列。例如[1, 1]其实就是"A1"单元格，[1, 2]其实就是"B1"单元格。

用字符串的形式来指代工作表和单元格，符合Excel的一般习惯，比较容易让业务人员读懂。用整数的形式来指代工作表和单元格，可以把整数保存在变量里，根据业务逻辑的需要随时变化。例如可以用[x, y]来指代单元格，这里的x和y可以是变量名，随着循环而依次递增，就可以依次把多个单元格都读出来，显然灵活性更好。具体用哪种形式，可以根据实际需要来选择。

“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UiBot每次读取一个单元格，既低效又麻烦。实际上，UiBot已经考虑到了读取一个区域的需求，提供了“读取区域”的命令，可以一次性的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。

输出到 ?

arrayRet ×

^ 必选

工作簿对象 ?

Exp objExcelWorkBook ×

工作表 ?

Exp Sheet1

区域 ?

Exp A1:B2

显示即返回 ?

Exp 是

图 79: 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“工作簿对象”和“工作表”，这两个属性表示需要读取哪个工作簿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"A1:B2", 表示读取的是从左上角A1单元格到右下角B2单元格, 共计2行2列, 4条数据。

当然, 除了用字符串之外, 还可以用一个“二维数组”来指代要读取的区域, 例如"A2:B6"可以写为[[2,1], [6,2]], 这里的几个整数都可以写为变量, 这样的话, 读取的范围就可以根据业务逻辑来变化了, 灵活性更好。当然, 如果您还不了解二维数组, 可以暂时先忽略这种指代方式, 先用字符串来指代就好。

“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。例如, 对于下图所示的表格, 我们要求UiBot读取"A1:F2"的区域, 并且在“读取区域”命令之后, 加入一条“输出调试信息”命令, 将arrayRet这个变量的值打印出来。



	A	B	C	D	E	F
1	刘备	关羽	张飞	赵云	马超	黄忠
2	20K	18K	15K	12K	10K	10K

图 80: 要读取的Excel表格内容

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, 在上图的例子中, 输出的结果是: [["刘备", "关羽", "张飞", "赵云", "马超", "黄忠"], ["20K", "18K", "15K", "12K", "10K", "10K"]]

对于编程的概念不熟的读者, 可能还不了解“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。UiBot提供了一系列的Excel写入命令来修改工作簿的内容。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, 可以看到这条命令的属性如下图所示:



图 81: 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，如果在普通模式下填写这个属性，会以文本格式写入Excel的单元格。如果切换到高级模式，还可以填写数值、字符串，变量或者表达式。

Excel写入类命令，还有一个很重要的属性叫作“立即保存”。如果这个属性选择“是”，那么写入操作会被立即保存，就好比我們手动修改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”属性就应该是一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。UiBot中还提供了“创建多维数组”命令，可以快速创建一维、二维甚至更高维的数组，以便写入。

在Excel操作完成后，建议使用UiBot的“关闭Excel工作簿”功能，把当前操作的Excel工作簿关掉。否则，即使UiBot的流程运行结束了，Excel仍然是打开状态的，还会消耗系统资源。特别是当我们在打开Excel工作簿时选择了“不可见”的时候，虽然Excel的界面被隐藏了，但其实一直还处于打开状态，不仅会消耗系统资源，还不太容易被发现。

5.2 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流程中经常会遇到的。

和Excel类似，用UiBot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UiBot打开一个Word文档。在UiBot Creator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 模拟数据.docx 文件。



图 82: 打开Word文档

接下来是“访问时密码”和“编辑时密码”两个属性。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空即可。

“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：

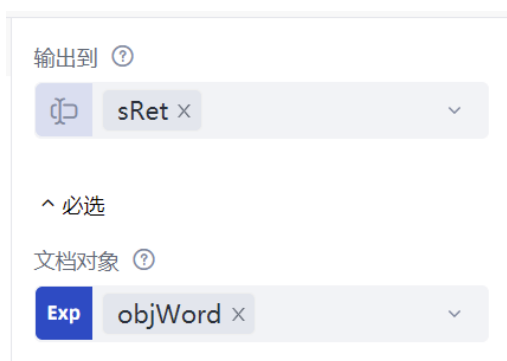


图 83: 读取Word文档

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的内容显示出来，运行后，可以看到如下结果：

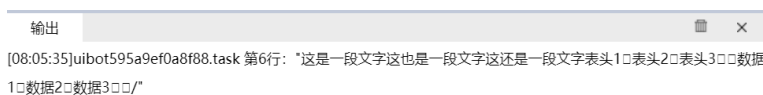


图 84: 读取Word文档的输出结果

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。



图 85: 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UiBot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。



图 86: 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。



图 87: 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。那如何解决这个问题呢？这里教大家一个技巧：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。

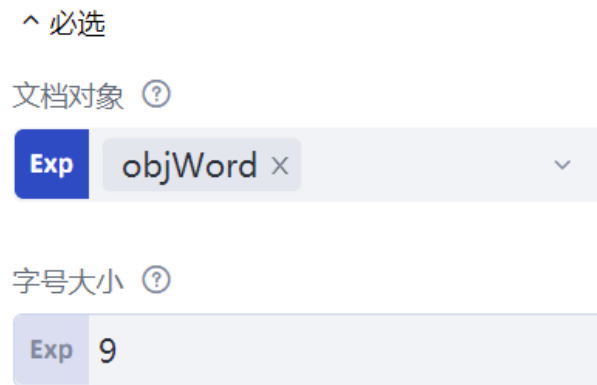


图 88: 设置文字大小

5.3 邮件客户端自动化

我们在日常工作中，经常需要发送或接收邮件。让RPA流程来自动发送或接受邮件，是很常见的需求。为了实现邮件的自动发送和接收，通常有两种方法：一种是直接通过SMTP/POP3/IMAP等邮件协议来实现，一种是通过邮件客户端来实现。前者不需要在计算机上安装任何客户端软件即可完成，但配置较为繁琐。后者需要依赖于邮件客户端软件，但相对比较简单。本章先介绍后者，即依赖邮件客户端的方式。UiBot支持两种常见的邮件客户端：Microsoft Outlook和IBM Notes。

Outlook是微软公司的主打邮件传输和协作客户端产品，也是Office软件套装的组件之一。下图是使用Outlook2019编写一封新邮件的界面展示：

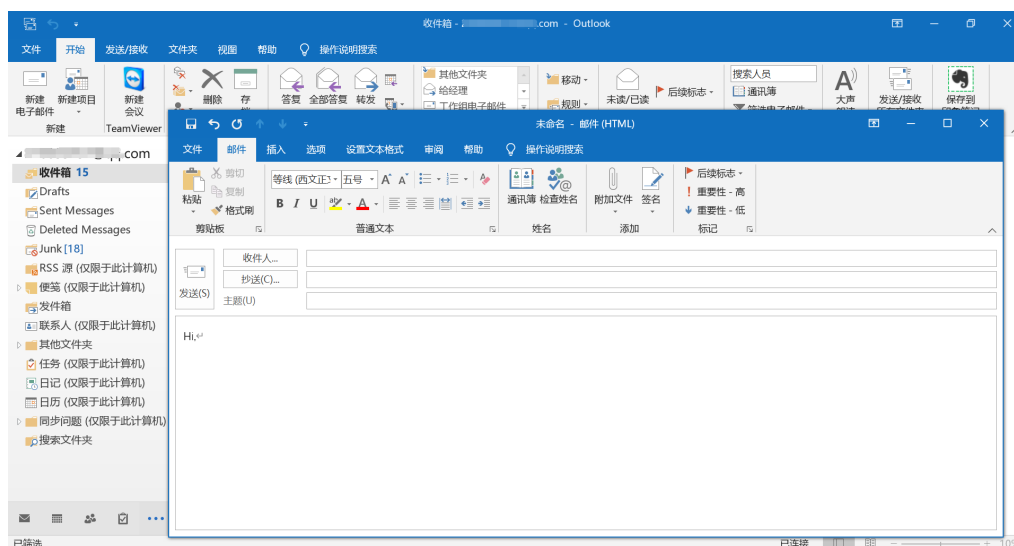


图 89: Outlook

使用UiBot自动化操作Outlook客户端，可直接使用“发送邮件”、“获取邮件列表”、“回复邮件”、“下载附件”这些命令，与用户平时在Outlook客户端上操作邮件的习惯基本相同，填写发件人、收件人、标题、正文、附件等信息即可，只是填写“发件人邮箱”、“邮箱地址”属性时，该邮箱地址一定事先在Outlook中绑定好(支持绑定多个邮箱地址)。

例如，在UiBot中插入一条“发送邮件”的命令，并对命令的属性进行适当的设置即可。这些设置和您人工发邮件时填写的内容几乎没有区别，非常简单。



图 90: 发送邮件

如果要收邮件，可以插入一条“获取邮件列表”的命令，并对命令的属性进行适当的设置。用一条命令可以收下多封邮件，“邮件数量”属性就是指定您需要收的邮件的数量，如果为0，代表收下邮箱中所有的邮件。



图 91: 收取邮件

另外，也支持比较低频的邮件操作，如“移动邮件”、“删除邮件”命令。需要注意的是，当前Outlook自动化命令主要适配Outlook的2010、2013、2016、2019版本。

同Outlook一样，UiBot还支持对IBM Notes 邮件客户端进行自动化操作，发送邮件、回复邮件、获取邮件、下载附件这些操作同样直接使用“发送邮件”、“回复邮件”、“获取邮件列表”、“下载附件”这几个命令就可以实现；前提条件一样，都需要事先绑定发件人邮箱，而不同点在于配置属性：IBM Notes 不需要填写发件人邮箱，但存在“密码模式”，所有自动化命令都要配置密码，包括“移动邮件”、“删除邮件”命令。下图以“发送邮件”为例进行展示。其他各个命令也比较类似，不再赘述。



图 92: 用IBM Notes发送邮件

UiBot在自动化操作IBM Notes的时候，对IBM Notes版本也存在一定要求，目前主要适配了9.0.1和11.1的中文版。IBM Notes的版本号在如下的界面中可以查到。

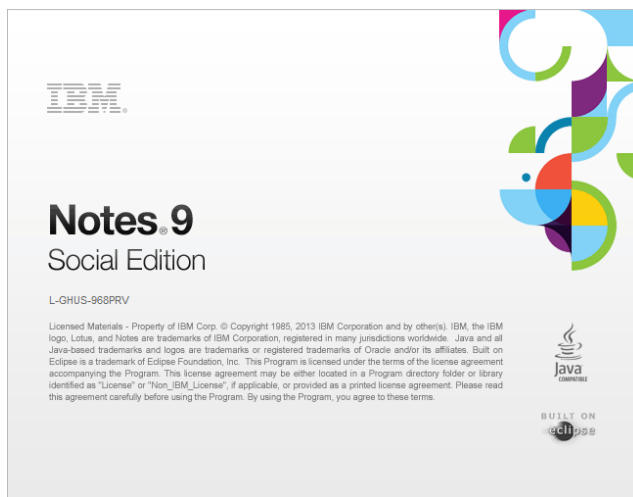


图 93: IBM Notes 9.0

5.4 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。



图 94: 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。UiBot目前支持IE浏览器、Google Chrome浏览器、火狐浏览器、UiBot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，UiBot自带浏览器是UiBot Creator 5.0版本后自带的浏览器。在这里，我们选择的是“UiBot Brower”，即UiBot自带的浏览器。相比其它三种浏览器，UiBot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、UiBot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、UiBot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UiBot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，UiBot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“UiBot”，并点击“百度一下”按钮，就可以得到“UiBot”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，这些功能将在后续教程“数据处理”中详细讲解，这里就不再展开。

5.5 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于数据库中。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。

输出到 [?]

objDatabase ×

^ 必选

数据库类型 [?]

Exp MySQL

数据库配置 [?]

Exp {\"host\": \"\", \"port\": \"3306\", \"...\"}

图 95: 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，UiBot目前支持MySQL、SQL Server、Oracle、Sqlite3、PostgreSQL共五种数据库类型。“数据库配置”属性描述了创建数据库对象时的一些关键信息，这段信息比较长，也不太容易看懂，不过没关系，点击右边的“纸和笔”按钮，会弹出一个窗口，显示“数据库配置”的更多属性，如下图所示：

数据库配置 ×

Host [?]

Exp 192.168.0.1

Port [?]

Exp 3306

User [?]

Exp root

Password [?]

Exp

Database [?]

Exp user

确定 取消

图 96: 数据库配置

“host”和“port”指的是数据库的IP地址和端口号，这里填写的是“192.168.0.1”和“3306”，表明数据库可以通过“192.168.0.1:3306”这个地址进行访问；“user”和“password”指的是访问数据库的用户名和密码；

“database”指的是我们连接的数据库的名称；“charset”指的是数据库的字符集，通常保持默认“utf8”即可。数据库的具体配置各有不同，以上信息的配置，可以询问您要访问的数据库的管理员。

当然，每种类型的数据库，其配置属性可能不完全相同，比如Oracle数据库，没有“database”参数，只有“sid”参数，但其含义是类似的。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。对于PostgreSQL数据库，“数据库配置”属性与MySQL比较略有不同，但当前仅支持关系型特性的自动化操作，PostgreSQL其他的现代特性暂不支持。

和其他的软件自动化命令类似，可以在“输出到”属性这里填写一个变量名，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。UiBot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。



图 97: 执行单SQL查询

最后切记，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。

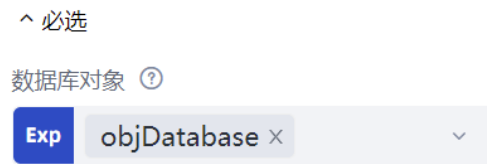


图 98: 关闭连接

6 逻辑控制

前文中我们讲过，一个流程块通常包含多条命令，在前面的例子中，流程块中的多条命令都是一条一条按顺序来执行的，比如一个流程块完成Excel数据写入的功能，依次执行了“打开Excel”、“读取单元格”、“保存Excel”、“关闭Excel”四条命令。通常，我们把这种顺序执行的流程结构叫做顺序结构。但是实际的RPA场景远比这种情况要复杂，本章介绍稍微复杂一点的流程结构，以及在UiBot中如何使用逻辑控制来实现这些复杂一点的流程结构。

6.1 条件分支

首先介绍的这种流程结构叫做条件分支，什么叫做条件分支呢，顾名思义，指的是流程结构运行到某一步骤时，按照一定的条件进行分支：当条件满足时，按照其中一条分支走下去；当条件不满足时，按照另一条分支走下去。

我们来看具体的命令用法。在UiBot Creator的命令列表中，选中“基本命令”并展开，再选中“语法词法”并打开，找到“条件分支”，用这条命令就可以建立一个条件分支。

可视化 源代码



图 99: 条件分支命令

在UiBot的命令组装区，可以清晰地看到“条件分支”的详细用法。条件满足时的分支处写着：如果“条件成立”，则，下方用虚框写着提示语此处可插入执行命令，我们在此处插入一条“输出调试信息”命令，这条命令输出内容“条件成立时，输出这条消息”；条件不满足的分支处写着：否则，下方用虚框写着提示语此处可插入执行命令，我们在此处插入一条“输出调试信息”命令，这条命令输出内容“条件不成立时，输出这条消息”。这个时候命令组装区变成这个样子：



图 100: 条件分支命令 — 添加输出调试信息

我们试着运行一下，果然出错了：



图 101: 条件分支命令 — 运行结果

为什么出错？因为这个时候，“条件分支”命令最重要的属性“判断表达式”，我们还根本没有填写，好吗！打开“条件分支”命令的属性区，“判断表达式”属性处，UiBot帮我们默认填了一个文字版的条件成立，但这仅仅是个提示罢了，我们要把条件成立这句话替换成真正的条件表达式。

^ 必选

判断表达式 ?



图 102: 条件分支命令 — 条件表达式

在这里，“判断表达式”属性是一个布尔类型的属性，其值只能是“真（True）”或者“假（False）”，这个值可以通过常量、变量或者表达式得到，这些概念我们暂时还没讲到，没关系，后面会详细讲解，现在我们只需要记住这里可以填写布尔类型的常量、变量或者表达式即可。出于演示的考虑，这里我们填写True。

^ 必选

判断表达式 ?

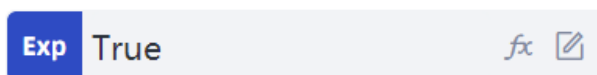


图 103: 条件分支命令 — 条件表达式为真

这个时候，我们发现命令组装区，条件满足时的分支处写着：如果 真，则，表明“判断表达式”属性已经生效。运行这条命令，得到正确结果，输出调试信息：“条件成立时，输出这条消息”。

需要说明的是，“条件分支”命令的两条分支，是两个命令块。在命令块中，根据需要，可以放置一条命令，也可以顺序放置多条命令，当然也可以一条命令都不放，空着。使用“条件分支”命令时，“条件不成立”这条分支不填写内容，也是一种常见的用法。

6.2 循环结构

我们再来介绍另一种重要的流程结构，叫做循环结构，什么叫做循环结构呢，顾名思义，指的是流程按照一定的规则循环执行。按照循环规则的不同，又可以分为计次循环、条件循环两种，遍历数组和遍历字典其实也是两种特殊的计次循环，但是由于现在我们还没讲到数组和字典，因此遍历数组和遍历字典也放到后面再讲。

6.2.1 计次循环

先来看看计次循环。在UiBot Creator的命令列表中，选中“基本命令”并展开，再选中“语法词法”并打开，找到“计次循环”，用这条命令就可以建立一个计次循环。将“计次循环”命令添加到命令组装区后，我们再在循环体内添加一条“输出调试信息”命令，这条命令会把“索引名称”*i*依次作为调试信息输出。

可视化 源代码



图 104: 计次循环

这里引出了“索引名称”的概念。我们打开“计次循环”命令的属性列表框可以看到，“计次循环”命令有四个属性：“索引名称”是用来计次的数值，这里用变量 *i* 表示，*i* 在循环体中也可以使用（上面的例子中我们就将 *i* 依次显示出来）；“初始值”和“结束值”标定了循环的范围，“步进”默认值为1，也可以修改为其它值。这三个值合起来的含义是：*i* 从“初始值”开始，每循环一次自动增加“步进”的值，直到大于“结束值”，循环才会结束。我们运行这条命令，可以看到，打印出0到10，循环一共执行了11次。

^ 必选

索引名称 [?]

初始值 [?]

Exp

结束值 [?]

Exp

步进 [?]

Exp

图 105: 计次循环的属性

6.2.2 条件循环

再来看看条件循环。在UiBot Creator的命令列表中，选中“基本命令”并展开，再选中“语法词法”并打开，找到“条件循环”，用这条命令就可以建立一个条件循环。将“条件循环”命令添加到命令组装区后，我们再在循环体内添加一条“输出调试信息”命令，这条命令输出内容"条件为真，继续循环"。

可视化 源代码

1	<input type="checkbox"/>	当 真 成立则循环执行操作
2	<input type="checkbox"/>	向调试窗口输出: 条件为真,继续循环

图 106: 条件循环

“条件循环”命令的属性区与“条件分支”一样，有且只有一个属性：“判断表达式”。“判断表达式”为真，循环才会执行，为了让循环执行起来，我们在“判断表达式”处填入True。

^ 必选

判断表达式 [?]

Exp

图 107: 条件循环的属性

执行“条件循环”命令，我们发现，会一直不停的输出字符串"条件为真，继续循环"，而不会自动停止。需要我们点击UiBot Creator工具栏的“停止”按钮，才能强行停止流程的执行。回到“条件循环”命令的定

义，所谓“条件循环”，指的是：满足一定条件时，将会循环执行某一语句块。相应的，如果条件不满足，将不会执行那个语句块。在刚才的示例中，我们为了让循环执行起来，我们在“判断表达式”属性处填写了一个固定的布尔值True，而这个值不会随着循环变化，因此“判断表达式”一直为真，循环也无休止地运行下去。

那怎么来解决这个问题呢？第一种方法，UiBot提供了多种跳出循环的命令，包括“继续循环”、“跳出循环”、“跳出返回”和“退出流程”等。这个我们接下来马上就会讲到；第二种方法，也是更加通用的做法，在“判断表达式”中填入一个表达式，最开始这个表达式的值为真，随着循环的进行，表达式的值不断发生变化，当循环达到某种状态时，表达式不再为真，这个时候循环就结束了。

我们来举个例子：首先定义一个变量a，并给这个变量赋初值为1；然后在“判断表达式”中填入 $a < 5$ ，一开始这个表达式是成立的（因为这个时候a等于1），循环开始执行；接着在循环中给a的值加上1；就这样，经过几次循环后，a的值不再小于5，循环随之退出。

同样需要说明的是，不管是“计次循环”还是“条件循环”，其循环体都是命令块。命令块中可以放置一条命令，也可以顺序放置多条命令，命令块中的命令也可以是“条件分支”命令或者“计次循环”、“条件循环”本身，即逻辑控制命令是可以嵌套的，这是一个非常重要的概念。

6.3 循环的跳出

我们在上一节说过，UiBot提供了多种跳出循环的命令，包括“继续循环”、“跳出循环”、“跳出返回”和“退出流程”等命令。其中有些命令不仅可以用以循环的跳出，甚至可以用于流程块和流程的退出。下面我们就分别来讲解一下。

首先是“继续循环”命令，所谓“继续循环”，指的是在执行循环体的过程中，不再执行本次循环，而是在终止本次循环后，跳回到循环体开始处，继续执行下一次循环。

可视化 源代码



图 108: 继续循环

其次是“跳出循环”命令，所谓“跳出循环”，指的是在执行循环体的过程中，不再执行循环命令，而是直接跳出循环体，继续执行循环语句后面的命令。

可视化 源代码



图 109: 跳出循环

再次是“跳出返回”命令，所谓“跳出返回”，指的是在执行循环体的过程中，不再执行循环命令，而是直接跳出所在的流程块，并返回retValue这个值。

可视化 源代码



图 110: 跳出返回

最后是“退出流程”命令，所谓“退出流程”，指的是在执行循环体的过程中，不再执行循环命令，而是直接退出流程，中止流程的执行。

可视化 源代码



图 111: 退出流程

需要注意的是：“跳出返回”命令和“退出流程”命令不仅可以用于循环体当中，也可以用于条件分支和顺序结构中，也就是说流程块的任何位置，只要有需要，都可以随时通过“跳出返回”命令和“退出流程”命令达到跳出本流程块和退出流程的目的。

7 流程和任务管理

我们在前面章节已经学习过，一般RPA平台至少会包含三个组成部分：开发工具、运行工具和控制中心，在UiBot中，这三个组成部分分别叫做UiBot Creator、UiBot Worker和UiBot Commander。

在上述章节中，我们只讲述了UiBot Creator的使用方法。经过学习之后，可能很多初学者也会在脑海中冒出一个问题：UiBot Creator已经足够强大了，能够完成绝大部分开发功能，当然也能运行流程。那为什么还需要用UiBot Worker和UiBot Commander？或者说，UiBot Worker和UiBot Commander到底能为我们带来哪些额外的价值？

要回答这个问题，还是要回到RPA机器人流程自动化的初衷和原始概念。我们说RPA是虚拟员工、是数字劳动力，是企业雇佣一些数字员工替代人类员工来完成一些机械重复而又繁琐的工作。那么问题来了：一家正规的企业雇佣一名员工，是否会不加管理而放任其随意工作？肯定不会！在企业中，对员工的工作普遍有以下几点要求：第一、任务的贯彻执行。这就要求员工接受上级领导的指令和任务，完成上级领导布置的任务，保证企业朝着同一个目标努力；第二、员工之间的配合。多个不同层级、不同部门的员工协调配合，共同完成一个复杂的任务；第三、监督和管理机制。在工作的过程中，需要有一定的机制进行监督和管理，当出现异常的时候需要及时处理。而上述几点功能，仅仅使用UiBot Creator是无法完成的，而这正是UiBot Worker和UiBot Commander的功能点和亮点所在。

因此，如果你只是用RPA完成一些个人事项，虽然这些事项也是一些机械重复而又繁琐的工作，但是既不是上级派发给你的，也不需要他人配合，更不需要监督和管理，即使出错也不是那么要紧，那么使用UiBot Creator就够了，在某些领域，例如游戏领域，我们甚至推荐使用按键精灵完成类似任务。而如果你是在一家企业里面实施RPA项目，那么UiBot Worker和UiBot Commander则是不可或缺的。本章将介绍这两者的使用方法。

7.1 RPA的两种工作模式

当RPA在企业里面使用的时候，通常需要用UiBot Creator开发流程，并完成调试和测试，之后，再由UiBot Worker来运行流程。因为UiBot Worker有两种工作模式，是UiBot Creator所不具备的。

第一种称之为“人机交互模式”，通常安装在您的桌面计算机上，以一个桌面工具的形式出现。能够根据您的需求，选择安装UiBot Creator编写好的流程，并且在您日常工作过程中，需要运行某个流程的时候，按下“运行”按钮开始运行。除此之外，还可以设置触发器，当满足一定条件时（例如到了指定的时间，或者收到了指定的邮件等），只要UiBot Worker还在运行，就会自动运行这个流程。由于是安装在桌面计算机上，因此，在运行的时候，人有可能还在计算机前面等候。可以在机器人运行的过程中进行必要的人机交互，比如机器人弹出一个对话框，由人来确认等等，如果需要停止机器人的运行，也可以直接在桌面计算机上进行操作。

第二种称之为“无人值守模式”，通常安装在专门运行RPA流程的计算机上，安装之后，只要进行了必要的设置，UiBot Worker就直接在后台运行了，连界面都没有。这台计算机可以放在桌面上，也可以直接部署在服务器机房，如果是在机房的话，甚至连键盘鼠标和显示器都没有。那要如何启动流程呢？可以通过UiBot Commander，向无人值守的UiBot Worker去派发任务。通过UiBot Commander派发任务的好处是：可以非常灵活地控制一批UiBot Worker，比如同时让多个UiBot Worker运行同一个流程。甚至可

以采用“抢单制”，派出去一批流程运行的任务，然后自动分配到空闲的UiBot Worker上。比如有100个任务，有10个UiBot Worker，那么可以先自动派出10个任务，哪个UiBot Worker运行完了，就自动再去取一个新的任务，直到100个任务全部做完为止。任务的运行结果和日志，也都在UiBot Commander上集中管控。

在有的资料中，把“人机交互模式”也叫作RDA（Robotic Desktop Automation），而把“无人值守模式”才叫作RPA。当然，名字叫什么不重要，关键是要搞清楚这两者之前的区别。“人机交互模式”适合于个人使用或者小规模使用，简单、快捷而又直接。“无人值守模式”适合大规模使用，在UiBot Commander上集中管控多个机器人，显然机器人的数量越多，越能体现出便利性。我们在UiBot Worker的社区版中，免费提供了一个“人机交互模式”的使用授权和一个“无人值守模式”的使用授权，您可以体验到两者的差异。如果需要更多的数量，可以购买商业版的授权。

7.2 流程管理

用UiBot Creator编写流程之后，为了在UiBot Worker中运行，通常采用的方法是将流程上传到UiBot Commander，然后再由UiBot Commander下发到UiBot Worker。这一过程大致如下图所示：

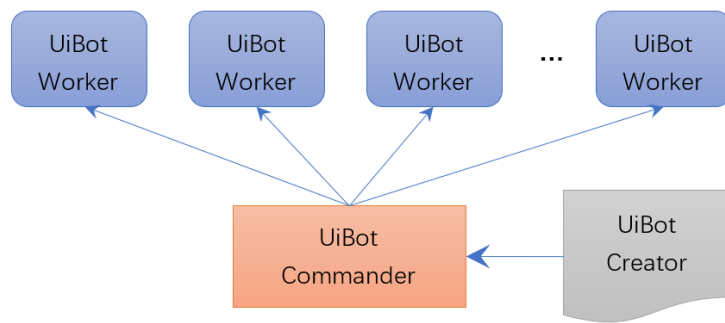


图 112: 流程在UiBot“三件套”内部的流转

我们以社区版的UiBot Commander为例。您可以在浏览器的地址栏中直接输入<https://commander.laiye.com>，也可以在使用UiBot Creator的任意时刻，找到右上方的“登录信息”，点击后，在弹出的对话框里面选择“前往Commander”。两者的效果是一样的。

UiBot Commander是B/S（Browser-Server）架构的软件，无需安装任何客户端，在浏览器中即可使用全部功能。打开UiBot Commander后，可以看到其主要功能都放置在左侧的菜单中，包含了设备管理、流程管理、任务管理等一系列功能。我们首先关注其“流程管理”功能，如下图所示：



图 113: UiBot Commander的界面及“流程管理”功能

为了方便说明，在您首次登录UiBot Commander的时候，我们已经预置了一个叫作“演示流程-UiBot自我介绍”的流程。当然，您一定还希望建立自己的流程。在右上方找到“新建”按钮并点击，在弹出的对话框中填写流程名称即可。如果有多人协作，推荐您再写一段更详细的流程说明，以便他人理解。如果担心流程运行时间太长，还可以设置一个“最大运行时长”，以分钟为单位，到了最大运行时长之后，无论流程有没有结束，都会自动结束。

在UiBot Commander中创建了一个流程之后，可以回到UiBot Creator，在流程图视图下点击工具栏里面的“发布”按钮，并选择“发布至Commander”，如图所示：



图 114: 发布流程UiBot Commander的界面及“流程管理”功能

此时，UiBot Creator中会弹出一个对话框。对话框中有一个名叫“选择流程”的下拉列表，其中会列出UiBot Commander上的所有流程。如果您使用的是社区版，这里应该会列出您刚才在UiBot Commander上新建的那个流程。这样选择之后，就把UiBot Creator中创建的流程，和UiBot Commander中创建的流程对应起来了。所以如果使用企业版的UiBot Commander，可能有多人协作，这个下拉框里的流程也会比较多，一定要仔细选择，避免张冠李戴。除了选择流程之外，还要给流程增加一个“版本号”，这个版本号的格式并无规定，可以是任意字符串，使用者方便区分就行。比如“1”或“1.0”或“1.0.0”都可以是版本号。当UiBot Creator多次发布一个流程到UiBot Commander的时候，UiBot Commander会保留每次发布的内容，并用“版本号”来区分。这样一来，如果一个流程的某个版本出了问题，还可以切换到其他版本。

发布完成后，还需要在UiBot Commander上启用刚才发布的这个版本。点击流程的名字，从右侧展开一个窗口，窗口里面显示了流程的概况。切换到“版本列表”标签页，可以看到从UiBot Creator上发布的每个版本。需要使用哪个版本，就打开这个版本后面的“开关”（我们称之为“启用”），这样一来，将来运行这个流程的时候，运行的就是刚才启用的这个版本。



图 115: 在UiBot Commander上启用流程的某个版本

有的读者可能会觉得这样的操作很繁琐。明明在UiBot Creator上已经发布了，还要在UiBot Commander里面再“启用”一次，好像有些多此一举。其实不然！因为在企业内部使用的时候，UiBot Creator和UiBot Commander可能并不是同一个人在操作。如果UiBot Creator的操作人员发布了一个新的版本，而UiBot Commander的操作人员毫不知情的话，那么流程实际上已经变了，UiBot Commander的操作人员还一无所知，这是有安全隐患的。所以，在UiBot Commander上的“启用”操作相当于是一个“审批”的过程，只有经过了审批，才会被真正启用。

启用后的流程，既可以在人机交互模式的UiBot Worker中运行，也可以在无人值守模式的UiBot Worker中运行。要在这两种模式下运行，您需要做的操作完全不同，下面逐个解释。

7.3 人机交互模式

如果您使用的是社区版的UiBot，安装完成后，会自动在桌面上生成一个UiBot Worker Community的图标。双击这个图标，就会以人机交互模式开始运行UiBot Worker。



图 116: 在人机交互模式的UiBot Worker中启动流程

当在UiBot Commander上新建了流程，且在UiBot Creator中发布了流程的至少一个版本，那么，在UiBot Worker的界面上就可以直接看到这个流程。在流程的右边，还有一个“运行”按钮（如下图红框所示），按下按钮，这个流程就开始运行了。如果在运行过程中希望停止，还可以长按热键Ctrl+F12。

运行后的结果可以在UiBot Worker界面左边的“运行记录”标签页下面看到，除了能看到每次运行的时间，运行结果等，后面还有两个小按钮，可以进一步查看运行的日志信息，以及运行过程中的屏幕录像。如下图：



图 117: 查看流程的运行记录

看起来比用UiBot Creator运行一个流程稍微多了点儿功能，但并没有本质的区别。实际上，仅仅是“运行流程”这一件事情，UiBot Worker还能玩出不少花样。

• 分身运行

大部分自动化流程在运行的时候，都需要占用鼠标和键盘的控制权，而人机交互模式的UiBot Worker通常部署在桌面计算机上，当流程运行时，计算机前的操作人员只能等待它运行完，再进行人工的工作。白白等待，浪费了时间。“分身运行”功能能够启动一个独立的窗口，并让自动化流程在这个窗口里面运行，即使窗口被遮挡了也不会影响。这样一来，流程运行的时候，还能同时进行人工的操作，两者共用一台计算机，数据共享，却互不干扰。进一步提高了工作效率。

启用分身运行之前，需要先安装分身运行的扩展程序。然后，在流程的“运行设置”对话框中，可以找到“分身运行”的选项，开启即可。



图 118: 开启“分身运行”的选项

开启了分身运行之后，在流程启动时，会弹出一个窗口，这个窗口里显示了一个独立的Windows桌面，

需要您用Windows账号密码登录。登录完成后，流程就会自动在这个独立的Windows桌面上运行了。

关于“分身运行”的具体功能介绍，请查看这一段视频。

- 触发器

除了手动运行流程之外，UiBot Worker还提供了“触发器”的功能，可以在满足一定条件的时候，有计划地、有选择性地执行流程。UiBot Worker提供了四种触发器：定时触发器、启动触发器、邮件触发器、API接口触发器。在UiBot Worker的界面上找到“触发器”标签页，并点击右上角的“新建”按钮，即可新建不同类型的触发器。

- 1) 定时触发器：可以设定流程的启动时间，包括设定单次运行、按日期、按周、按月等等。设定完成后，UiBot Worker会持续地检查时间，发现到了启动时间，就会开始运行指定的流程。
- 2) 启动触发器：这个触发器相对比较简单。设定完成后，在UiBot Worker每次启动的时候，都会自动运行指定的流程。
- 3) 邮件触发器：可以设定一个或一组邮箱地址，并且设定一个检测周期（默认是5分钟）。设定完成后，UiBot Worker会按照固定的时间周期，持续地检查邮箱，发现收到了符合条件的邮件，就会开始运行指定的流程。
- 4) API接口触发器：设定这个触发器之后，UiBot Worker会在本机上指定的TCP端口监听。可以通过HTTP请求，控制UiBot Worker运行某个流程。

- 流程组

在某些场景下，多个流程之间可能存在依赖关系，例如，流程B需要流程A先完成，才能继续运行。UiBot Worker提供了流程编组的功能。所谓“流程编组”，指的是将两个或多个有依赖关系的流程放置到一个编组中，编组中的流程顺序执行，一个流程执行完，再顺序执行下一个流程。当然，还可以设定更为复杂的组合关系，比如流程A运行M次以后，流程B再运行N次。或者如果流程A运行失败了，流程B还要不要继续运行，等等。通过这些设置，可以把多个流程像积木块一样搭在一起，灵活应对各种场景的需求。

如下图所示，在人机交互模式的UiBot Worker的界面上方，选择“流程组”这个标签页，并且点击“添加”按钮。然后，在弹出的对话框中，选择流程进行编组即可，被选入流程组的各个流程均可设置运行次数，还可以通过拖动改变他们的运行次序。

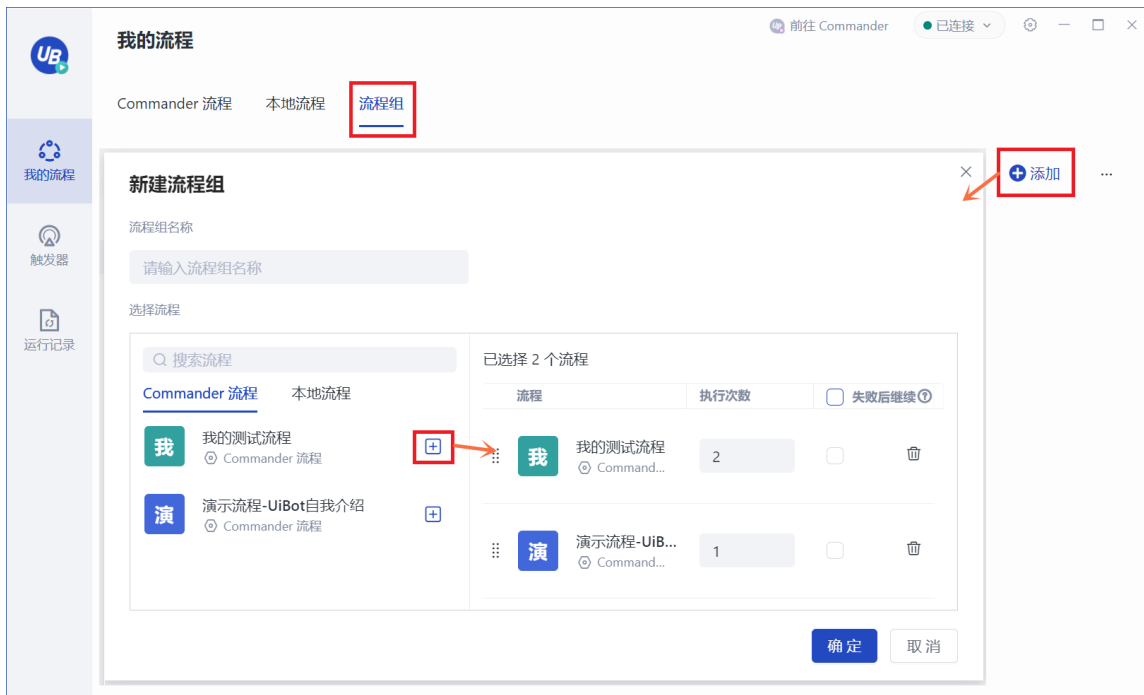


图 119: 创建流程组

当流程组创建完成后，可以把流程编组当作普通流程看待，也就是说，流程编组既可以立即运行，也可以通过计划任务安排运行，还可以支持分身运行。

• 快速开始

从UiBot 6.0版本开始，安装之后，还会生成一个名为UiBot Widget的图标，我们称之为“快速开始”。顾名思义，这相当于是人机交互模式下的一个流程的快速启动工具。

其实，人机交互模式下的UiBot Worker已经能够非常方便快捷地启动流程了，但有的流程可能是需要在人的日常工作中时常启动的。比如呼叫中心里面的话务员，每接到一个电话，可能就需要运行一个RPA流程来帮助他自动录入相关信息。此时，UiBot Worker的窗口仍然显得太大，难免干扰话务员的正常工作。但如果每次运行完流程就把UiBot Worker关掉，需要需要流程的时候再启动，又显得非常繁琐。

“快速开始”工具就是针对上述场景设计的，这个工具启动速度快，占用系统资源少，平时以可以半透明的方式置于Windows桌面一角，几乎不影响日常工作。当需要运行RPA流程的时候，鼠标移上去点击一下即可。“快速开始”工具里面总共可以容纳最多8个流程的运行按钮，如果您日常工作中用的RPA流程不超过8个，还可以把空余的按钮用来运行一些其他的常用工具。如下图所示，除了两个流程的运行按钮之外，还把运行Windows记事本、Windows画图工具、Windows截图工具和百度搜索的入口都各自放置在一个按钮上了。能有效提高工作人员的日常效率。



图 120: “快速开始”工具

“快速开始”工具几乎不需要任何设置，但它和UiBot Worker是联动的，UiBot Worker里面有哪些流程，这个工具里就会自动加入哪些流程。所以，在初次使用之前，可能仍然需要打开UiBot Worker，对流程进行必要的配置，比如刷新UiBot Commander上的流程，设置是否以分身的方式运行，等等。另外，“快速开始”工具上已经预置了四个常用工具（Windows记事本、Windows画图工具等）。如果您还希望预置更多的工具，或者不需要这些预置工具，目前还需要手动修改安装目录下的Widget.TurnTable.config文件。这个文件是JSON格式的，内容比较简单，有IT经验的读者很容易编写，因此本文不再赘述。未来我们也会将其设置功能加入到软件界面中。

7.4 无人值守模式

在无人值守模式下，UiBot Worker的工作方式和操作方式都和人机交互模式完全不同。我们可以在一台计算机上建立多个无人值守的UiBot Worker（这种方式称为“高密度部署”，一台计算机上可以部署的UiBot Worker数量没有上限，仅取决于这台计算机的性能），它们的工作不会互相干扰。并且，这些UiBot Worker都以后台服务的方式存在，根本没有界面。如果要让它们运行流程，所有的操作都在UiBot Commander上进行。

所以，在无人值守模式下，我们需要进行如下的配置工作：

- 把安装了UiBot，且要运行无人值守模式的UiBot Worker的计算机，纳入到UiBot Commander的管理范围中；
- 使用UiBot Commander，在被管理的计算机上建立UiBot Worker的实例。如前文所述，一台计算机上可以建立多个UiBot Worker的实例，相当于可以有多个机器人同时工作，且互不干扰。

我们先来学习如何对运行无人值守模式的UiBot Worker的计算机进行配置。配置完成后，就不需要再去操作运行UiBot Worker的计算机了，甚至这些计算机重新启动也没有关系。后续操作都在UiBot Commander上完成。

在计算机上安装UiBot后，会出现一个名为“UiBot Worker Hub”的图标，这是无人值守模式的配置程序。启动它，可以看到如下的界面：



图 121: 使用UiBot Worker Hub将设备加入UiBot Commander

这里需要输入密钥。密钥是什么呢？它是由UiBot Commander生成，让这台计算机和UiBot Commander能够实现身份互认的一个“密码”。可以把UiBot Commander比作您家里的WiFi路由器，路由器上有一个WiFi密码，任何要加入这个WiFi网络的设备，只要输入这个密码就行。所以，我们需要到UiBot Commander上去寻找这个密码。

在UiBot Commander上，可以为所有要加入的计算机设置同样的密钥，也可以为每个要加入的计算机设置一个独立的密钥。前者更方便，后者更安全。篇幅所限，本文仅以前者为例，后者的操作方式可以举一反三。

打开UiBot Commander，在左边找到并选中“设备”标签页，点击右上角的“通用密钥”按钮，在弹出的对话框中就显示了当前的密钥。将它填入到UiBot Worker Hub的密钥框中，并选择“立即激活”。

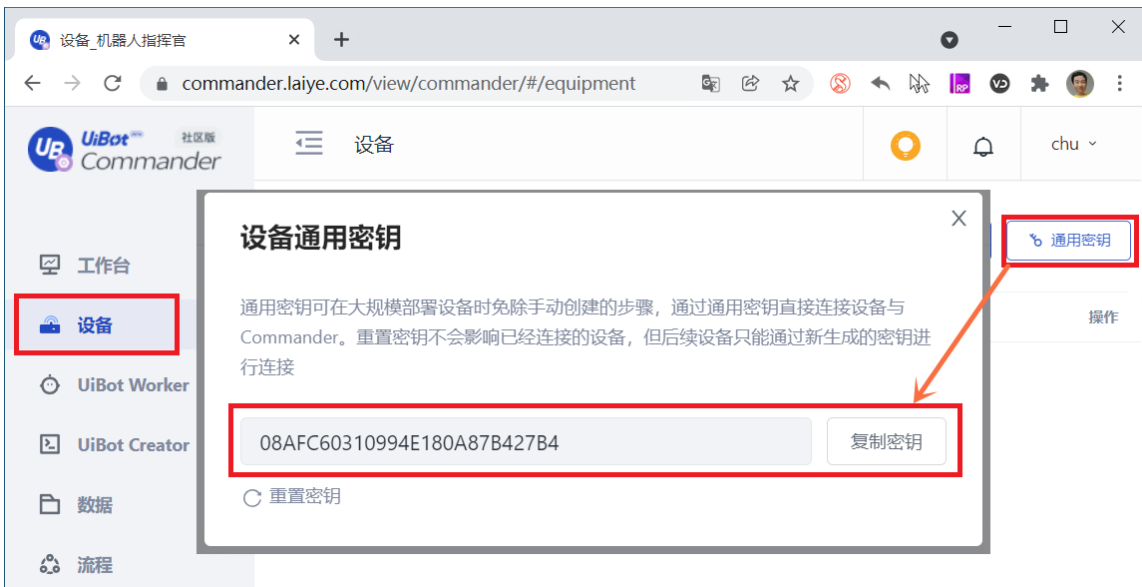


图 122: 在UiBot Commander中进行设备管理并找到密钥

以上激活步骤相当于向UiBot Commander发出了一个连接申请，在UiBot Commander的“设备”标签页中也能看到当前待处理的连接申请，当然，也可以选择同意或拒绝该申请。如果同意，就能看到新增了一台设备，与此同时，UiBot Worker Hub的界面上也会显示已经和UiBot Commander建立了连接。

接下来的操作基本上都是在UiBot Commander上完成了。首先，需要在刚刚连接的这台计算机上建立起一个或多个UiBot Worker的实例。只需要在UiBot Commander下面的“UiBot Worker”标签页下选择“无人值守”，并选择“新建Worker”，按照弹出对话框的提示，分别输入UiBot Worker的实例的名称、基于哪个设备创建实例、本地账户和密码等信息，如下图所示：

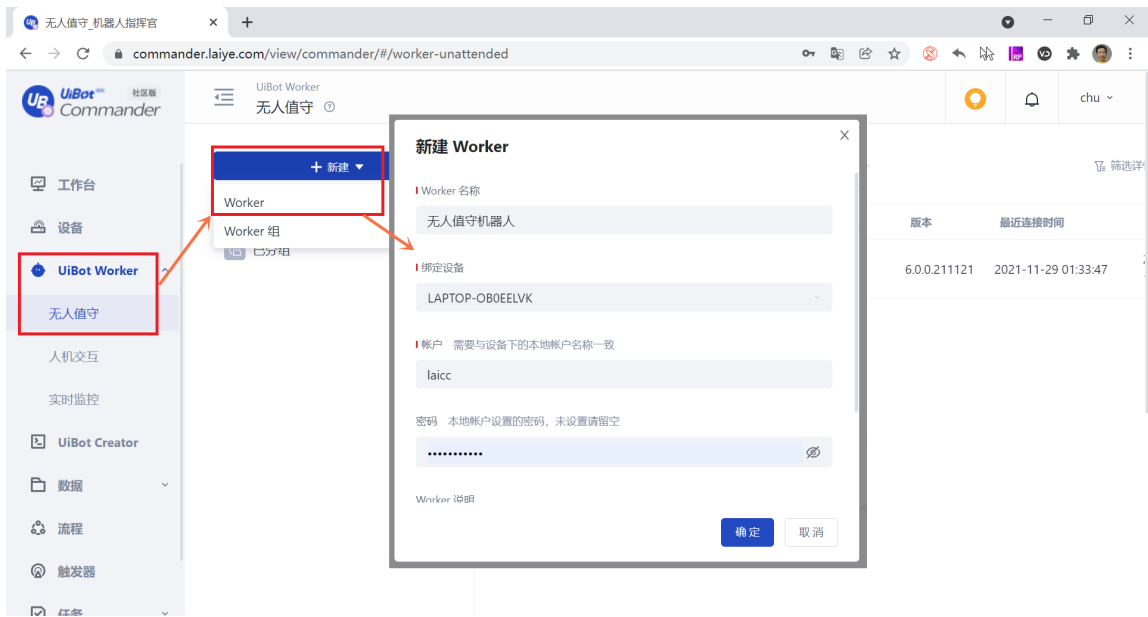


图 123: 通过UiBot Commander新建UiBot Worker的实例

注意：这里的“本地账户”是指Windows的账户，“密码”也是指这个账户对应的Windows密码。通过以下方法，可以很容易地看到当前的Windows都有哪些账户：在Windows开始菜单中找到“命令提示符”并打开，输入命令net user，在横线下列出的就是所有可用的本地账户了，如下图所示：

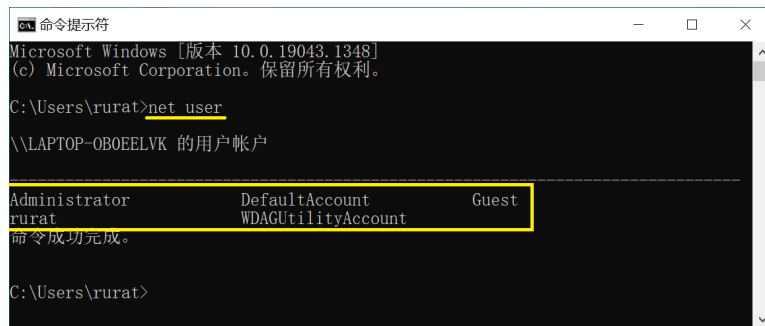


图 124: 通过“命令提示符”查看本机上所有的Windows账户

把某个Windows账户和对应的密码填写到“本地账户”和“密码”栏中之后，新建的UiBot Worker会以后台服务的方式，准备接收UiBot Commander发来的命令。当UiBot Commander命令它运行某个流程的时

候，即使UiBot Worker Hub程序已经关闭了，即使当时Windows刚刚重新启动完成，还没有任何账户在登录的时候，UiBot Worker也会使用您预设的Windows账户和密码，自动登录Windows，然后再进行自动化操作。从而做到真正的“无人值守”。

下面我们来实际尝试一下，在UiBot Commander中发起一个运行流程的命令。我们把运行中的流程称为一个“任务”（显然，使用一个流程可以运行多个“任务”），因此，在UiBot Commander的“任务-个人任务”标签页中，选择“新建”按钮，建立起一个任务，也就相当于是运行流程了。在“新建”任务时，首先需要选择流程，其实也就是指定要运行哪个流程；其次选择“部门”，是指这个任务可以被某个部门的用户看到；然后选择“Worker组”或“指定Worker”，前者是指在一组UiBot Worker中，随机选取一个空闲的UiBot Worker来“抢单”，后者则是指派特定的UiBot Worker来运行这个流程。

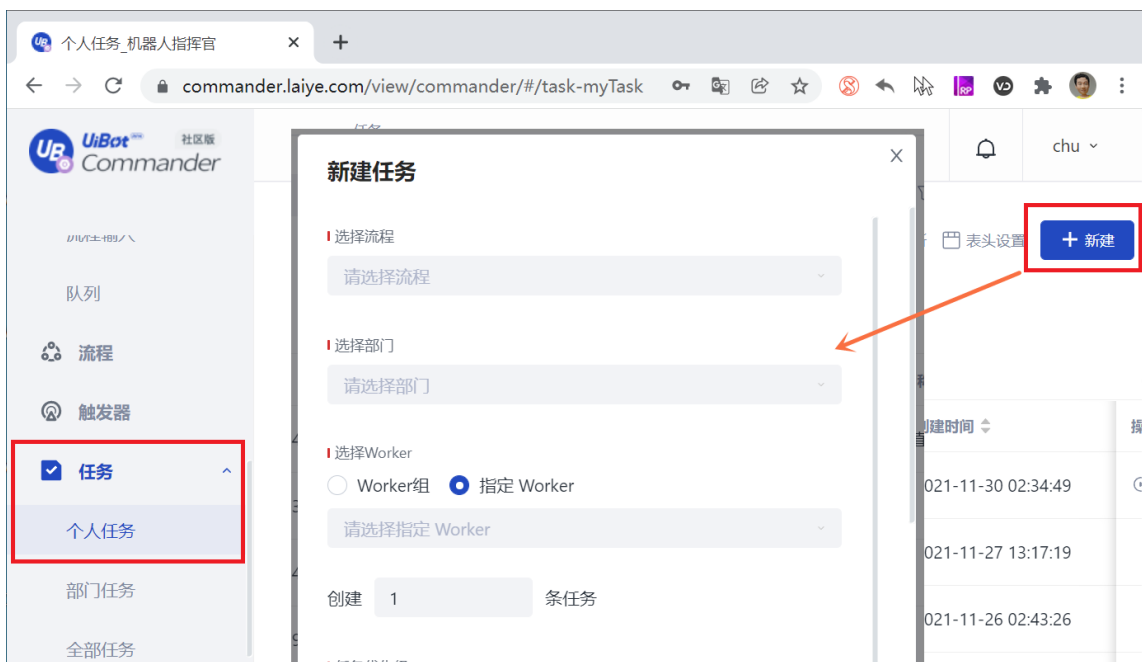


图 125: 新建一个任务

按上述方式新建任务后，流程会马上开始运行。当然，还可以选择“触发器”功能，在指定的时间自动建立任务，并运行流程。具体用法显而易见，本文不再赘述。

无论是立即建立任务，还是在触发器中自动建立任务，所有待运行、正在运行、已运行完成的任务都会在“任务”标签页中一览显示。对于已运行完成的任务，无论运行结果是成功还是失败，都可以通过点击任务所在行，并在弹出的窗口中查看任务运行过程中的日志。如果使用企业版的UiBot，还可以在编写流程时加入“上传屏幕截图”命令，并且在这里看到运行过程中的截图。通过日志和截图等信息，在流程运行失败的时候，可以比较方便地定位到具体原因。

另外，在无人值守模式下，我们不能像在人机交互模式下那样，亲眼看到UiBot Worker的运行状况，可能会觉得心里不踏实。如果使用企业版的UiBot，可以在“UiBot Worker”标签页中找到“实时监控”功能，像观看监控室里面的摄像头画面一样，一览当前正在运行的所有UiBot Worker的画面，当然，也可以放大其中某个画面进行查看，甚至对其进行远程控制，等等。有兴趣的读者，可以通过这一视频观看实际演示。

7.5 机器人的协同

无论是人机交互模式，还是无人值守模式，多个UiBot Worker（或者简称之为RPA机器人）在运行流程的时候，彼此之间还可以传递数据。善用这一机制，可以实现多个RPA机器人之间的协同工作。比如，一个机器人专门收取邮件，并下载附件里的票据，另一个机器人专门把票据内容读出来，填写到业务系统里面去。

对于上述场景，可能有的读者会有疑虑：明明可以把收邮件、下载票据、读取票据内容、填写业务系统放在一个流程里面，用一个机器人去运行就好了，为什么要拆成多个流程，还要让多个机器人分别去运行呢？如果工作量不大，用一个流程当然是最简单也最清晰的。但如果工作量很大，比如每天要处理上万份票据，就需要多个机器人来并行处理了。更合理的方式是把整个工作拆分为几个步骤，每个步骤配比不同数量的机器人，比如收邮件的速度比较快，可以少配一些机器人，读取票据内容的速度比较慢，需要多配一些机器人。

无论是把多个任务分配给多个机器人，还是把任务分成不同阶段，不同的机器人来做不同阶段，都需要涉及到机器人之间的协同。UiBot提供了“数据队列”的机制，可以方便地在多个机器人之间传递数据，进而实现协同工作。一个数据队列就相当于是一组数据放在UiBot Commander上排队，有队头和队尾。UiBot Creator和UiBot Worker可以把数据添加到队尾，也可以从队头取走一个数据。目前的数据队列还不支持“插队”、“查看队头数据但不取走”或者“从队列中间取走数据”等复杂操作，但简单的设计也能符合大多数的需求，并且大大降低了开发的难度。

为了使用数据队列，首先打开UiBot Commander，在左边的标签页中选择“数据-队列”，然后选择“新建队列”，设置一个队列名称，一个队列就建好了，如下图所示。当然，我们还可以新建多个队列，每个队列中存放不同类型的数据。



图 126: 新建一个数据队列

对于建好的数据队列，还要对其进行“授权”操作。如果您使用的是社区版的UiBot，“授权”操作并不重要，因为都是您自己在使用。但如果是企业版，需要认真设置，究竟以哪个用户身份登录的UiBot Cre-

ator和UiBot Worker可以使用这个数据队列。

授权完成后，即可在UiBot Creator中设计流程，实现数据队列的“放入”和“取出”操作。所谓的“放入”操作，是指把一个数据排到队尾，而所谓的“取出”操作，是指从队头取出一条数据，如果队列中没有数据，则取出的值是Null即空值。这两个操作在UiBot Creator中都有对应的命令，命令的位置如下图所示：



图 127: 数据队列的相关命令

假设已经在UiBot Commander上创建了名为“我的队列”的数据队列，那么下图中的两条命令分别是把字符串"我的数据"放入队列，以及从队列中取出一条数据。可以选中一条命令，并按命令右侧的三角形“运行”按钮，单独运行这一命令，以方便测试。先测试第一条命令，单独运行它，貌似运行后并不能直观地看到结果，但如果我们刷新一下UiBot Commander的界面，就会看到队列中有“未消费”的数据，这里的“未消费”是指数据被放入了队列，但还没有被取走，因此会暂存在UiBot Commander的队列中，而一旦被取走后，就变成了“已消费”的数据。单独运行第二条命令，一方面在UiBot Creator上会显示：被取出的就是我们刚才放入的那条数据，另一方面也可以在UiBot Commander上看到：数据从“未消费”变成了“已消费”。



图 128: 数据队列的放入和取出

利用数据队列的机制，可以实现机器人之间的协同。比如，我们先设置一个所有机器人都能访问到的公共网盘，然后由M个机器人去收邮件，下载附件里的票据文件，保存在这个网盘上，并把文件的完整地址放入数据队列。再由N个机器人去数据队列里取到票据文件的地址，拿到文件并进行识别和填报。如果数据队列里的数据越堆积越多，说明N的取值小了，可以适当增加N的数量，让多个机器人能高效地协同工作。

8 结束语

恭喜您！您已经完成了UiBot开发者初级教程的学习！

通过初级教程的学习，您已经掌握了UiBot最基础的概念和最基本的操作，已经能够编写最简易的流程。快去UiBot官网免费参加UiBot开发者认证，检验一下自己的学习成果吧！

当然，如果您想更进一步深入学习UiBot，我们也为您准备好了UiBot开发者中高级教程，您将学习到UB编程语言、数据处理、网络和系统、人工智能、命令扩展等更多实用和高级的功能，继续加油吧！

附录：编程基础知识

在阅读本文时，我们假设您有任意一门编程语言的最基本的经验，至少了解数据类型、变量、条件判断等基本概念。如果连这些概念都没有，我们将在本章对此进行简单的介绍。UiBot需要的编程基础非常少，也非常容易学，所以，不必担心，耐心读完、理解本章所讲的概念，就已经足够了。

当然，如果您已有编程基础，那么本章的内容完全可以不用阅读。

我们首先来看一张Excel表格。这张表格中的内容是完全虚构的，我们只是用它来解释下面的几个重要概念。

	A	B	C	D
1	订单号	顾客姓名	订单数量	销售额
2	3	李鹏晨	6	261.54
3	6	王勇民	2	6
4	32	姚文文	26	2808.08
5	35	高亮平	30	288.56
6	36	张国华	46	2484.7455
7	65	李丹	32	3812.73
8	66	谢浩谦	41	108.15
9	69	何春梅	42	1186.06

图 129: 虚构的Excel表格

8.1 数据

RPA的主要工作通常就是在处理各种数据，什么是数据呢？我们设想有一张Excel表格，里面的很多格子已经填写了内容，这些内容都是数据。数据是计算机和人类之间交换的信息。

实际上，数据还可以细分为结构化数据和非结构化数据。像这种整整齐齐的写在一个个格子中的，显然是结构化数据。我们需要接触到的大多数数据也都是结构化数据，所以用这个表格来理解数据的概念就够了。而像图片、声音、视频、网页这些大部分都是非结构化数据，这里就不展开讲了。

8.2 数据类型

在一般的编程语言中，都会把数据分为若干种不同的类型，UiBot常见的数据类型包含数值型、字符串型、布尔型、空值型、复合型等等。在初级教程中，学习除了复合型之外的几种类型就够了，复合型放在中级教程中学习。

数值型就是数字，可能是整数，也可能包含小数位（在计算机中一般称为浮点数），比如实例表格中的“订单数量”、“销售额”等等；

字符串型通常是一串文字，通常会用一对双引号（"）或一对单引号（'）包起来，以示区别；字符串中使用反斜杠来表示一些特殊字符，例如 `\t` 代表制表符，`\n` 代表换行，`\'` 代表单引号，`\"` 代表双引号，`\\` 代表反斜杠本身等。字符串中间可以直接换行，换行符也会作为字符串的一部分。也可以用前后各三个单引号（'''）来表示一个字符串，这种字符串被称为长字符串。在长字符串中，可以直接写回车符、单引号和双引号，无需用 `\n`、`\'` 或者 `\"`。

布尔型只有“真”和“假”两个值，当然也经常被称为“是”和“否”、“True”和“False”等，内涵都是一样的。空值型的值总是 `Null`，不区分大小写。

如何区分这几个数据类型呢？一般来说，数值型是可以加减乘除的；字符串通常只会连接，而没有其他运算；布尔型通常只有“与、或、非”等逻辑运算。比如表中“顾客姓名”一列的数据，做加减乘除和逻辑运算都是没有意义的，所以应该是字符串类型，写的时候要加一对双引号，如“李鹏晨”。

表中“订单号”一列的数据，本质上都是数字，可以加减乘除，但他们的加减乘除并无意义。所以既可以按数值型处理，也可以按字符串型处理，设计者可以根据需要酌情考虑。

8.3 变量

在上面的Excel表格中，每个数据都是保存在一个小格子当中的。而且，Excel给每个小格子都设定了一个名字，比如261.54这个数据所占用的格子，就被称为D2。这个格子里面的数据可能会改变，比如销售额可能会发生变化，但是，这个格子的名字，D2，是不变的。我们在Excel中只要取D2这个格子里面的数据，就可以取得其中最新的值。

“变量”是编程中一个常见的概念，和Excel中的格子一样，变量也相当于是一个格子，里面可以存放数据。变量也有名字，可以通过名字取得变量中保存的数据，也可以通过名字对变量“赋值”，也就是设置变量中保存的数据。

Excel中的格子命名都是“字母+数字”的形式，而编程语言中的变量命名会灵活很多，可以是一个很长的英文单词，或者用下划线连接起来的好几个单词，除了第一个字符外，后面还可以使用0-9的数字。有的编程语言，包括UiBot所使用的编程语言，还可以用汉字来命名变量。一般推荐把变量的命名设置的略长一些，最好是有意义的单词或词组，而不是像D2这样的“代号”。这主要是为了让我们在阅读的时候看得更清晰，对程序的运行并没有影响。UiBot变量名不区分大小写。

UiBot中的变量是动态类型的，无需再定义的时候声明变量的类型，即变量的值和类型都可以在运行过程中动态改变。这也符合一般脚本语言如Python、Lua、JavaScript的习惯。

定义变量名的方式是：`Dim 变量名` 再定义变量名的同时可以给变量赋值一个初始值：`Dim 变量名 = 值` 想要定义多个变量的话，可以这样定义：`Dim 变量名=值, 变量名1 Dim 变量名=值, 变量名=值` 同理，想要定义一个常量就可以这样定义：`Const 常量名=值, 常量名=值`

8.4 表达式

变量和变量之间、或者变量和固定的数据之间，都可以进行运算，它们运算的算式被称为“表达式”。由于变量的值可能会发生变化，所以表达式的值也可能会发生变化。在编程语言中写一个表达式以后，只要当运行到了表达式所在的这一行，才会去根据变量里面保存的数据，去计算表达式的值。

比如，`x + 2` 就是一个表达式，当我们不能确定x的值的时候，就不能确定这个表达式的值。如果在运行到这一行的时候，发现x的值是3，那么`x + 2`的值就是5。

当然，有的运算是没有意义的，比如我们对一个字符串做除法，就是没有意义的。但由于我们在书写表达式的时候，变量的值还没有确定，可能无法马上确定这个表达式有问题。到运行到这一行的时候，计算机才发现表达式有错，无法继续运行下去了，通常会报错并退出。

8.5 条件判断

在编写一段程序的时候，我们通常会一行一行的去写。在程序运行的时候，通常也会按照从上到下的顺序，一行一行的运行。当然，这种运行方式是不够灵活的，我们常常希望能在运行的时候判断某个条件，然后根据条件，决定是否要执行某一段语句。这就是条件判断语句。

条件判断语句在不同的编程语言中有不同的写法，但其大致形式通常都是一样的：

```
如果 <表达式> 则
    语句1
    语句2
条件判断结束
```

其含义是，在运行到“如果”那一行的时候，会计算其中的表达式的值。这个表达式的值应该是布尔类型，如果不是，通常会自动转换为布尔类型。计算机会根据这个表达式的值，来决定要不要运行被“如果”和“条件判断结束”所包夹的语句，也就是例子中的语句1和语句2。只有当这个表达式的值为“真”的时候，才会运行它们，否则，会跳过它们。

我们在程序中会经常遇到条件判断语句，比如如下的程序：

```
发送邮件
如果 发邮件没有成功 则
    给用户报告没有成功
条件判断结束
```

只要正确的填写“如果”这一行中的表达式，使其在“发邮件没有成功”的时候为真，在“发邮件成功”的时候为假，就能达到我们的目的。反之，通常条件判断语句没有写好，其实也都是表达式没有设定好。在这个例子里面，通常“发送邮件”语句会给一个变量赋值，告诉我们是否发送成功，我们只需要把这个变量置入“如果”这一行的表达式中，即可奏效。

8.6 循环

循环语句和条件判断语句的形式比较接近，通常是：

```
当 <表达式> 的时候循环
    语句1
    语句2
循环结束
```

和条件判断语句类似，在运行到“当XXX的时候循环”时，也会先计算其中的表达式的值，如果表达式的值为“假”，会跳过这个循环，直接运行“循环结束”后面的语句。但和条件判断语句最大的不同是，如果表达式的值为“真”，会在运行完后面的语句1和语句2以后，再次跳回“当XXX的时候循环”这一行，重新判断表达式的值。

这样一来，就可以用循环语句，让计算机来做重复性的工作了。比如如下的程序：

```
发送邮件  
当 邮件没有发送成功 的时候循环  
    再尝试发送邮件  
循环结束
```

当邮件没有发送成功的时候，这个程序会反复尝试，直到邮件发送成功为止。当然，和条件判断语句类似，循环语句里面最关键的还是如何正确的设置这个表达式。如果设置的不好，表达式始终为“假”，则有可能程序在这里一直循环，不会再往下运行，也不能结束，这种情况叫“死循环”。