

## 27 | 如何创建AI智能体管理笔记和收集信息（一）

月影 · 跟月影学前端智能体开发



你好，我是月影。

从这一节课开始，我们进入新的单元，将讨论如何借助 AI 来全面提升个人能力。

在这个时代，AI 给我们带来的是全新的机会，不仅仅是狭义的工作机会和工作能力，还包括个人成长。作为开发者，我们和不懂技术的普通人相比，有一个得天独厚的优势，那就是我们不仅能够使用现成的 AI 工具，还能够利用技术，为自己的学习和成长打造最适合的效率工具，从而节省学习时间和加速个人成长。

这节课，我们先从将 AI 用于知识的获取和收集讲起，通过一个智能收藏助手的例子来看看 AI 能够如何帮助我们成长。

### 基于 AI 的智能收藏助手

我们先看一下具体产品形态，它是一个智能体，作用是帮我们整理网页、转写英文资料、记笔记以及管理资料。



这个智能收藏助手有好几个功能，分别是：

当你发一个网址给助手，它会将网址保存起来。

当你发一个英文网站给助手并让它转写，它会将转写成中文笔记再保存。

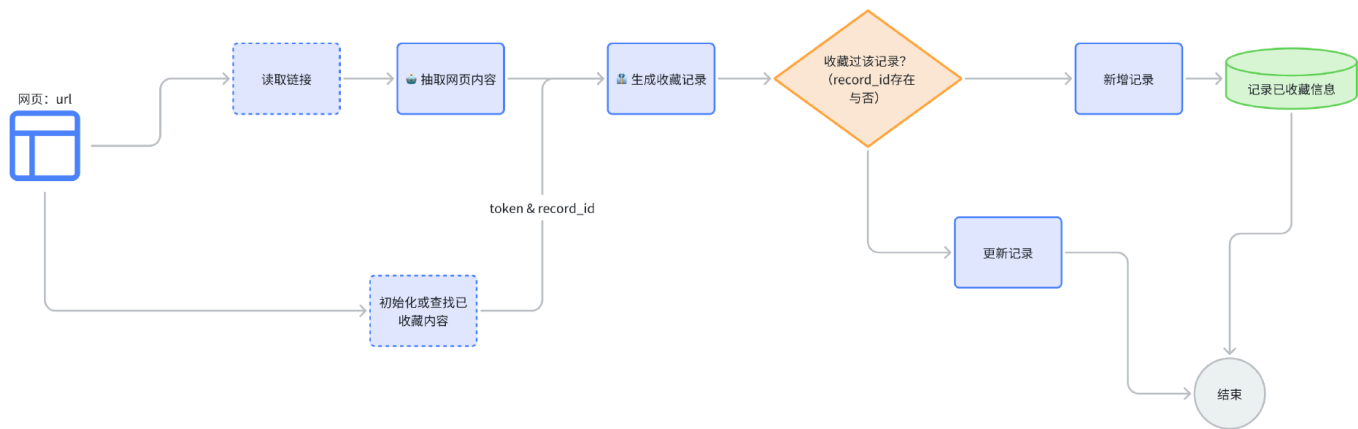
当你想记录一段笔记时，可以将内容发给它，内容可以是文本、图片或语音，助手都能将之记录下来。

助手会给这些内容打上标签，所以你可以让助手帮你从收藏中整理某些类型的内容并输出。

接下来我们一一来看这些内容的具体实现。

## 收藏网址

接下来，我们实现收藏网址功能的工作流，这个具体逻辑有点复杂，我们先来拆解一下：



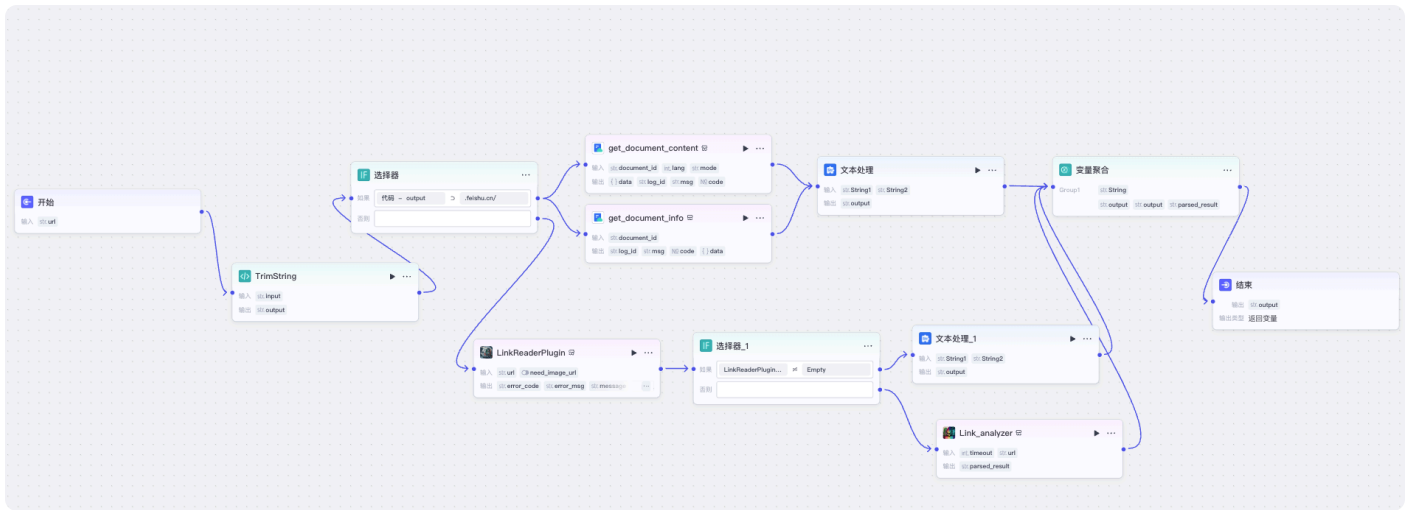
看上面的流程图，我们首先读需要取链接并抽取整理网页内容。同时，由于收藏夹需要去重内容，而且存放收藏内容的数据表格在第一次使用也需要初始化，所以我们还要调用初始化或查找已收藏内容的子流程。接着，我们将整理好的网页内容以及查找到的已收藏 record\_id（如有）结合生成收藏记录。然后通过已收藏 record\_id 判断是否收藏过该记录，如果已收藏过，则更新记录，否则新建一条记录，并更新已收藏数据。最后结束收藏流程。

注意在这个过程里面，一共有两个子流程，我们先来分别实现一下。

## 链接读取子流程

首先我们需要一个读取网页链接的子流程，这是因为虽然 Coze 提供了读取链接的插件，但是因为一些网站的限制，这个插件不一定能很好地工作，另外如果读取的资源是飞书文档，我们有更好的方式获取内容。

因此，这里我实现了一个增强的读取网页链接的子流程 **LinkReadPro**，它的工作流逻辑如下图所示。



首先，在开始节点中，我们设置输入参数 url。



接着我们在这里做一个细节上的处理，因为我发现有时候智能体发给工作流的 url 参数前后会多余空白符号，这个可以通过代码节点处理掉。

The screenshot shows a workflow editor with a 'TrimString' node. The node's input is 'str: input' and its output is 'str: output'. It is connected to an 'IF 选择器' (If Selector) node. The 'IF' node has a condition '代码 - output' and a value '.feishu.cn/'. The 'TrimString' node's configuration panel is open on the right, showing the following details:

- 输入 (Input):** Variable name 'input', Variable value 'str: 开始 - url'.
- 代码 (Code):** A code editor with the following content:

```
1 // 在这里，您可以通过 'params' 获取节点中的输入变量，并通
2 // 'params' 和 'ret' 已经被正确地注入到环境中
3 // 下面是一个示例，获取节点输入中参数名为 'input' 的值:
4 // const input = params.input;
5 // 下面是一个示例，输出一个包含多种数据类型的 'ret' 对象:
6 // const ret = { "name": '小明', "hobbies": ["看书",
7
8 async function main({ params }: Args): Promise<Outp
9 // 构建输出对象
10 const ret = {
```
- 输出 (Output):** Variable name 'output', Variable type 'str: String'.
- 异常处理 (Exception Handling):** A section for handling exceptions.

具体代码逻辑如下：

```
1 async function main({ params }: Args): Promise<Output> {
2     // 构建输出对象
3     const ret = {
4         output: params.input.trim()
5     };
6
7     return ret;
8 }
```

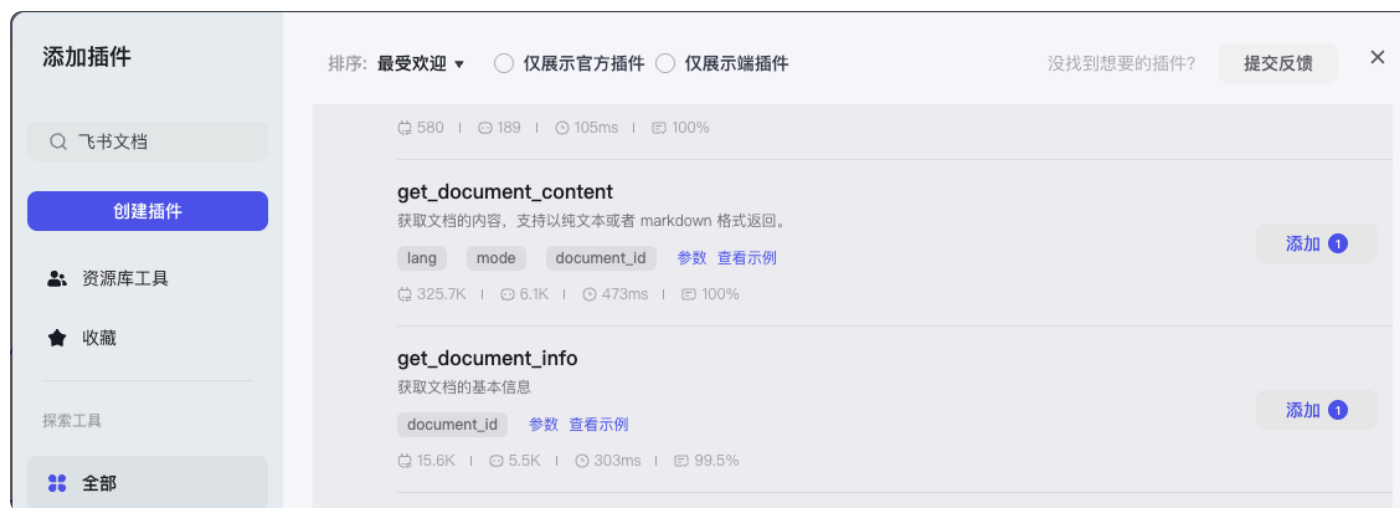
复制代码

接着是一个条件分支，我们通过 url 中的域名来判断是飞书文档还是其他链接。

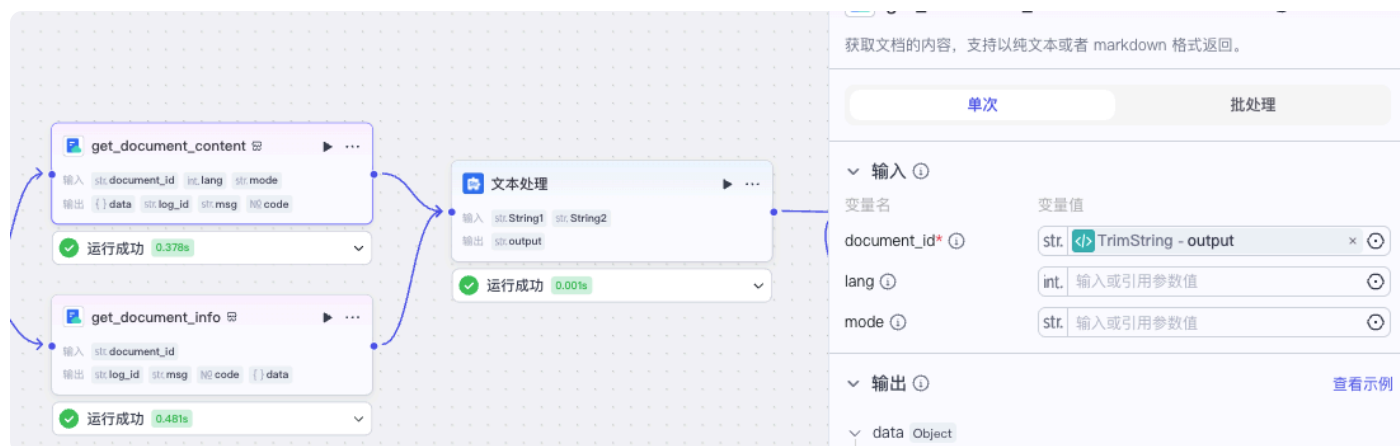


当网址包含 `*.feishu.cn/docx/` 或 `*.feishu.cn/wiki/` 或 `*.larkoffice.com/docx/` 或 `*.larkoffice.com/wiki/` 时，说明该内容是飞书文档，此时走飞书文档内容读取的分支。否则说明是普通链接，走普通链接读取分支。

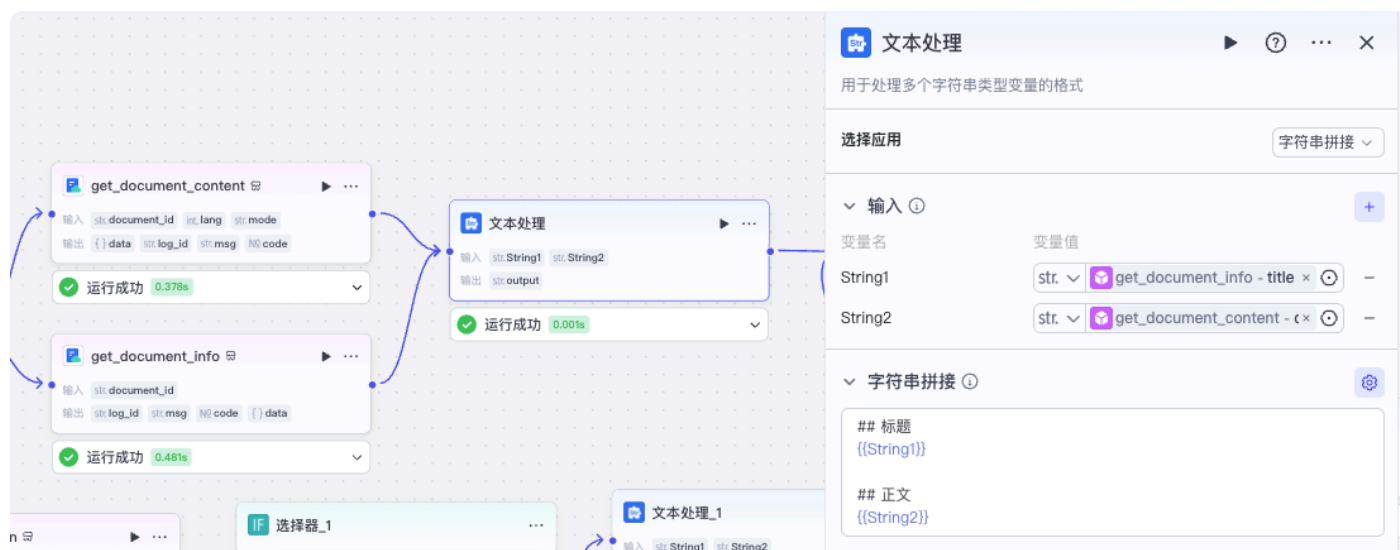
接下来，我们先看飞书文档的读取，我们添加插件，搜飞书文档，选择官方飞书文档插件中的 `get_document_content` 和 `get_document_info` 插件，分别获取文档正文和包括文档标题在内的文档信息。



我们先把去掉首尾空白的文档 url 作为 document\_id 分别传入两个飞书插件。



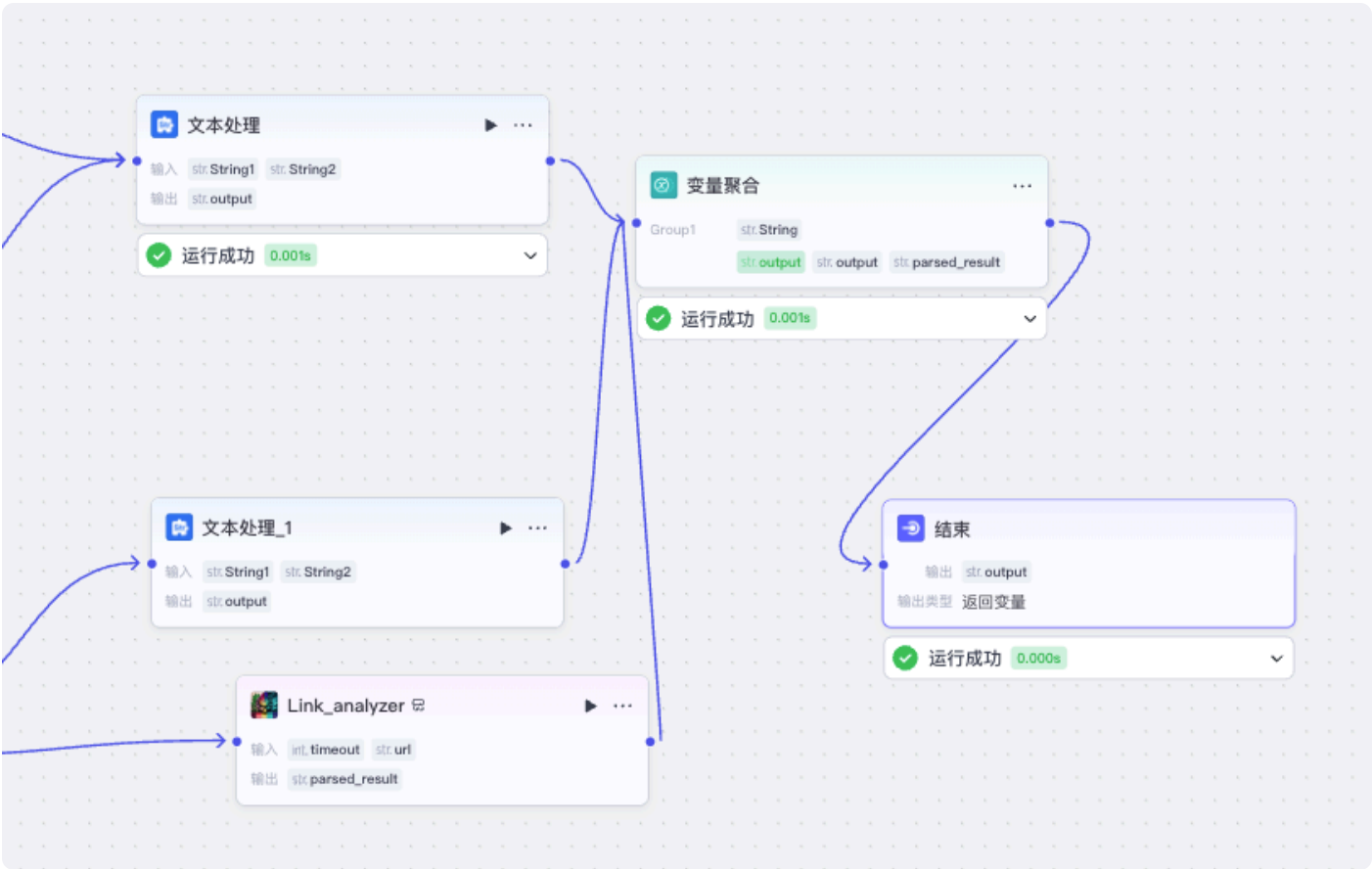
然后进行文本拼接处理。



接下来，我们处理普通链接的分支。



我们先尝试用官方的 LinkReaderPlugin 去获取链接内容，有时候它获取不到某个网页的内容，此时，我们再尝试用 Link\_analyzer 这个第三方插件去获取内容，这样可以增加成功率。



最后我们将两个分支的三个变量聚合起来，输出结果，这样就完成了链接读取的子工作流。

### 初始化或查找已收藏内容子流程

接下来我们实现第二个子工作流，这个工作流负责初始化收藏表以及根据 url 查找已收藏内容，防止重复收藏。

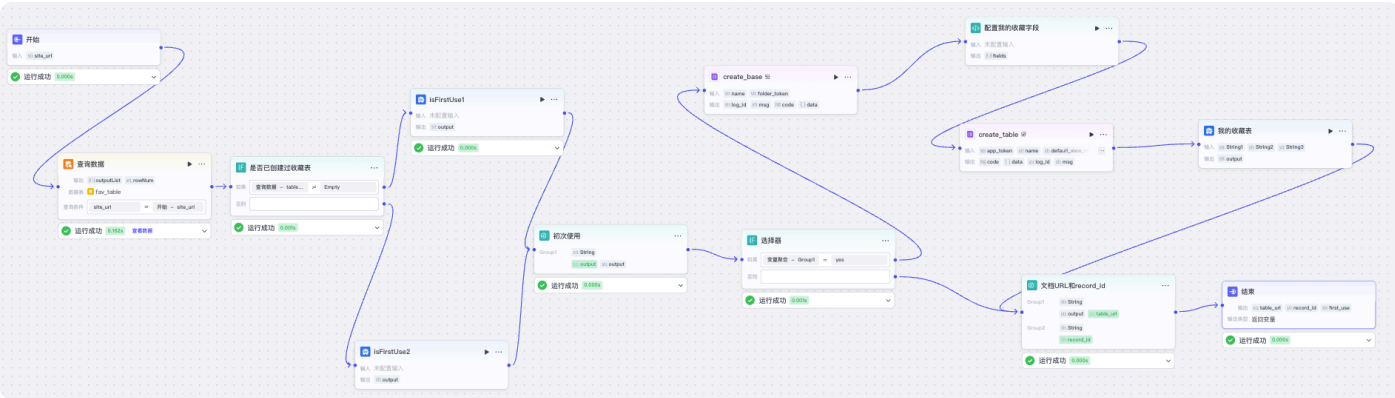


首先我们创建一个数据库表 fav\_table，但是我们只用这个表来记录以收藏的 url，真正的收藏信息我们不用数据库表来保存，而是使用飞书多维表格来实现。

之所以这么做，是因为飞书多维表格对使用者更为友好，而且还天然支持各种视图，方便我们直接管理收藏内容。但是飞书多维表格毕竟不是数据库，所以查询能力弱一些，因此将飞书多维表格和数据库结合，能同时结合两者的优点。

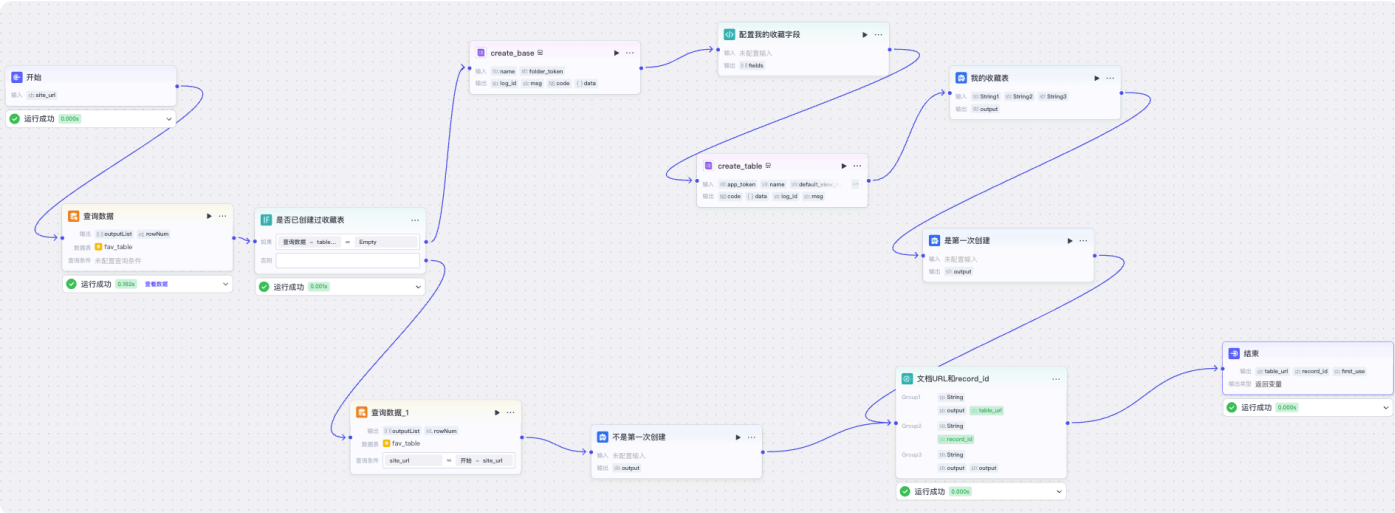
我们创建一个新的工作流 AI\_Favorites\_Init\_or\_Updated，功能是初始化或检查已收藏网址。

它的流程如下：



看起来流程稍微有点复杂，不过不着急，我们一步一步来。

首先要创建数据库表 fav\_table。



添加三个字段分别如下。

- table\_url：自动创建的多维表格文档 URL。
- record\_id：创建的多维表格中收藏记录的 ID，用于准确更新数据。
- site\_url：收藏网址，在数据库表中每个用户下唯一。

接着我们给开始节点设置输入变量 site\_url，然后不用查询条件，先查一下数据库表，看 table\_url 是否是空的。

开始

输入 site\_url

运行成功 0.000s

查询数据

输出 outputList int rowNum

数据表 fav\_table

查询条件 未配置查询条件

运行成功 0.162s 查看数据

查询数据

从表获取数据，用户可定义查询条件、选择列等，输出符合条件的数据

数据表

fav\_table

id uuid bstudio\_create\_time +3

查询字段

字段名

table\_url String

查询条件

排序方式

查询上限

1

输出

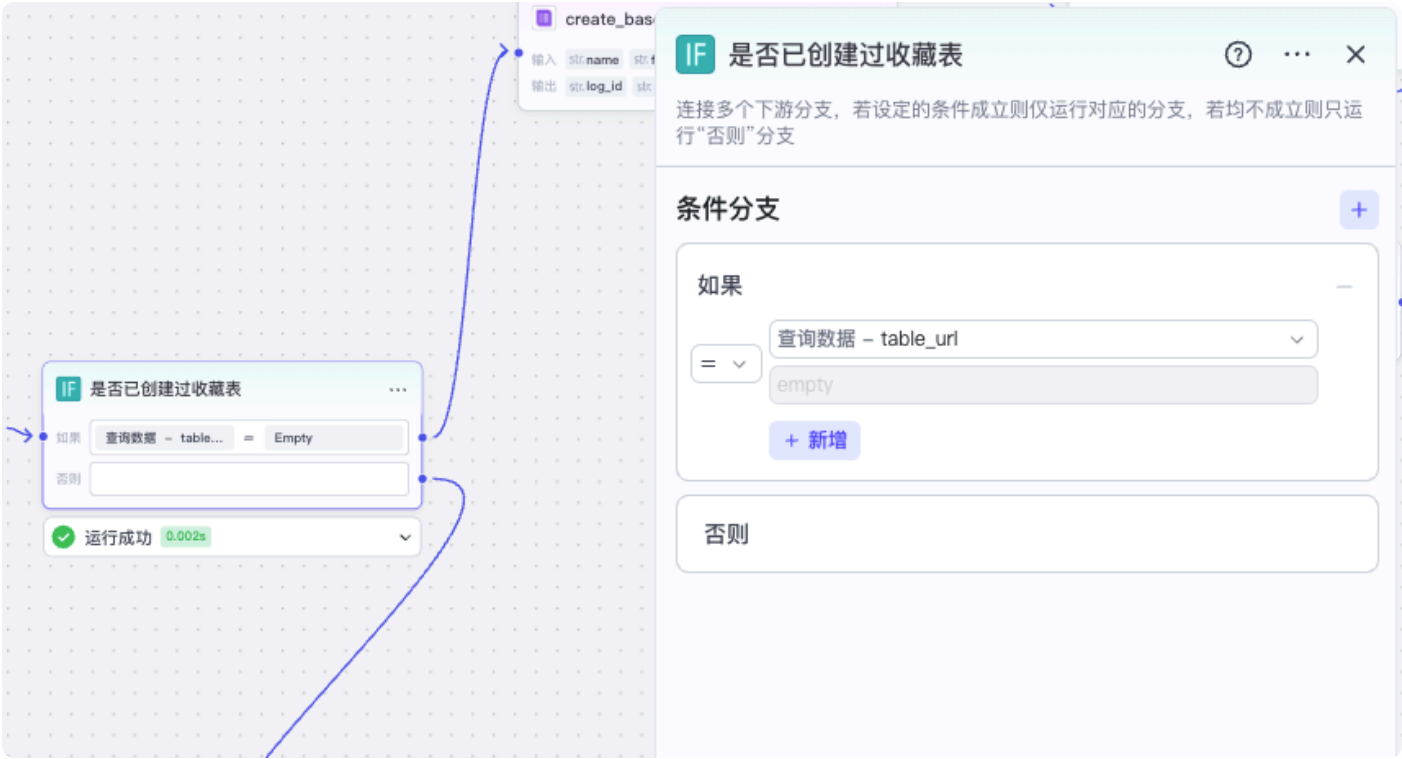
变量名 变量类型

outputList Array<Object>

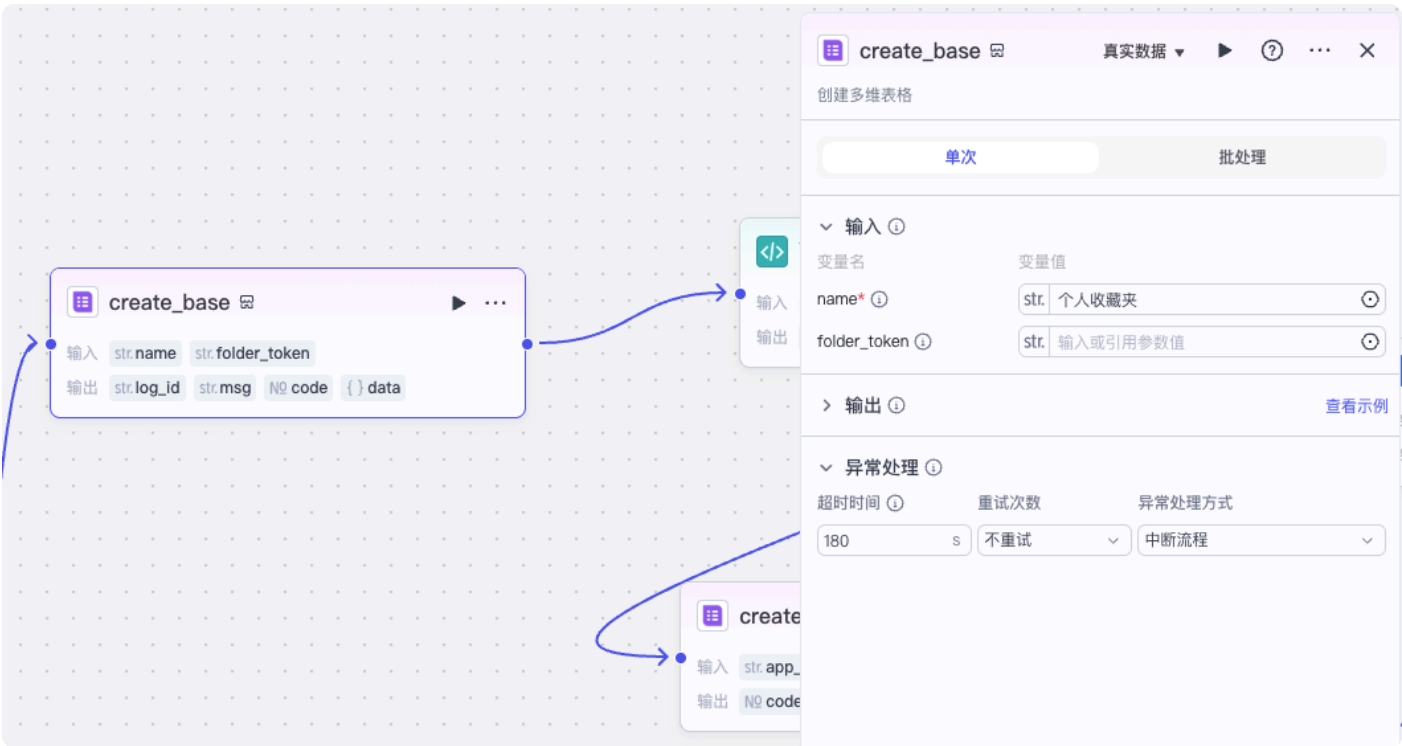
table\_url str. String

这里我们将查询条件设置为空，查询上限设置为 1，然后查询数据，如果内容不存在，说明该用户是第一次使用收藏助手，需要创建多维表格，否则说明已经创建过表格。

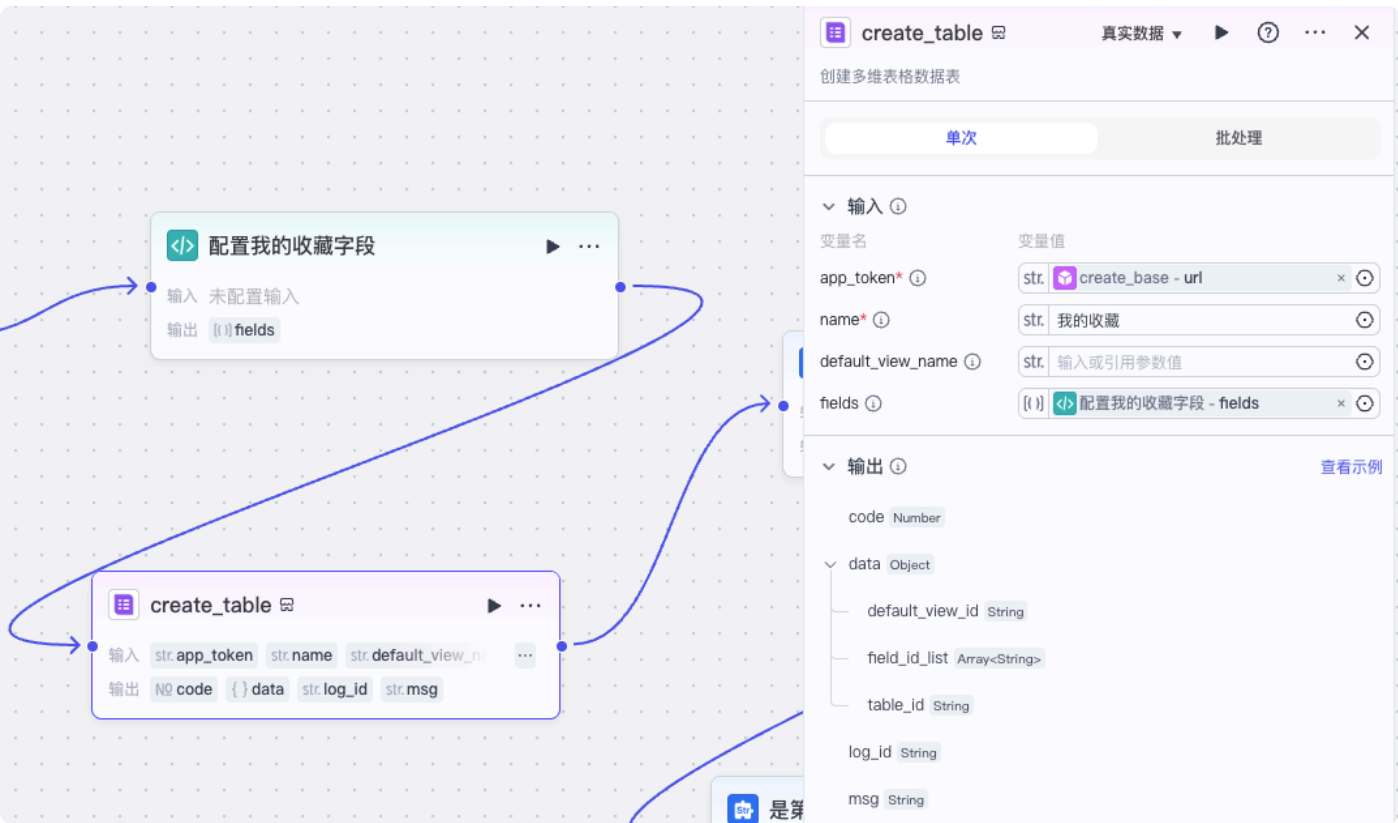
接着我们根据是否返回 table\_url，走两个分支。



如果 table\_url 为空，那么我们先通过多维表格官方插件的 **create\_base** 创建一个多维表格文件。



然后通过代码节点配置多维表格收藏字段，并在多维表格文件中用官方插件 **create\_table** 创建一个多维表格。



代码节点配置收藏字段的代码如下。

复制代码

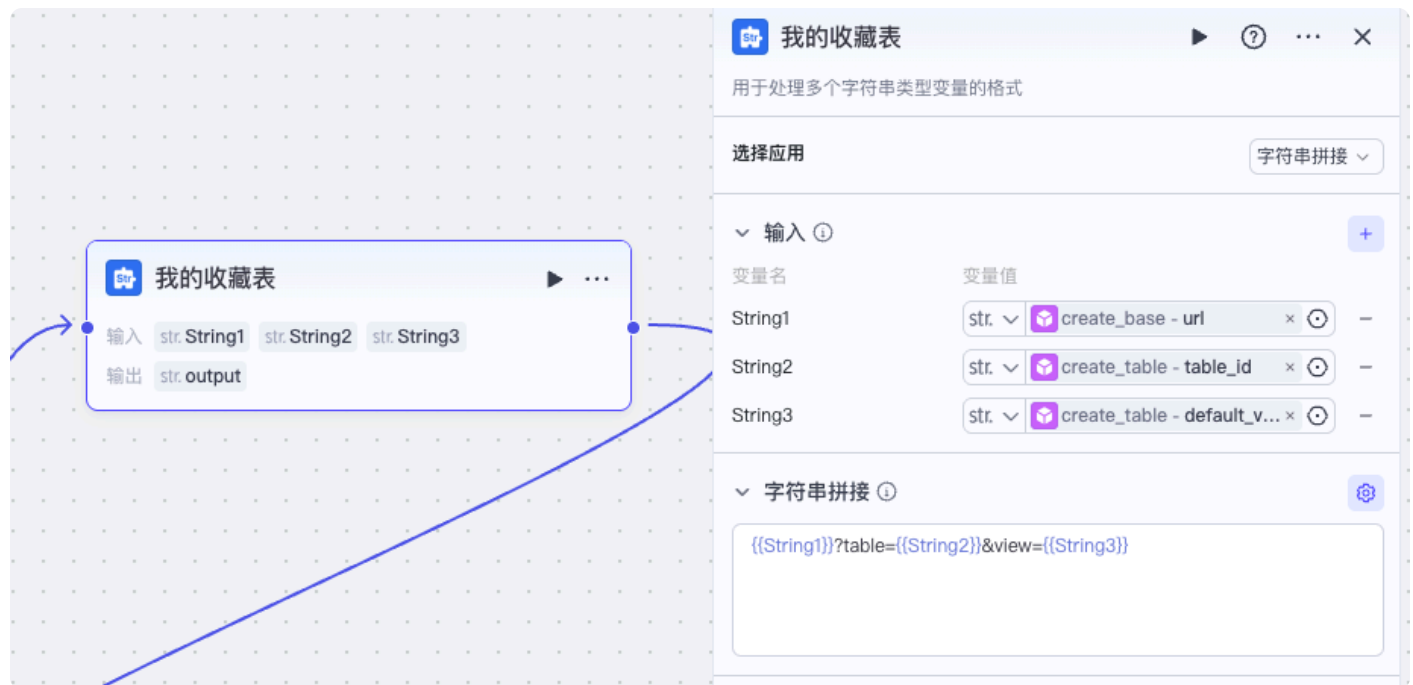
```
1 async function main({ params }: Args): Promise<Output> {
2   // 构建输出对象
3   const ret = {
4     fields: [
5       {
6         "field_name": "内容",
7         "type": 15 // 超链接
8       },
9       {
10        "field_name": "摘要",
11        "type": 1 // 多行文本
12      },
13      {
14        "field_name": "平台",
15        "type": 3 // 单选
16      },
17      {
18        "field_name": "标签",
```

```

19         "type": 4 // 多选
20     },
21     {
22         "field_name": "收藏时间",
23         "type": 5 // 日期 (时间戳)
24     },
25     {
26         "field_name": "状态",
27         "type": 3 // 单选
28     },
29 ]
30 };
31
32 return ret;
33 }

```

接着我们就可以用文本处理节点把收藏表的完整 url 拼接出来。



The screenshot shows a workflow editor interface. On the left, a node titled "我的收藏夹" (My Favorites) is connected to an input and an output. The input is labeled "输入" and has three variables: "str.String1", "str.String2", and "str.String3". The output is labeled "输出" and has one variable: "str.output".

On the right, the configuration panel for the "我的收藏夹" node is displayed. It has a title bar with a play button, a help icon, and a close button. Below the title bar, there is a subtitle "用于处理多个字符串类型变量的格式" (Format for processing multiple string type variables). The panel is divided into two main sections: "选择应用" (Select Application) and "字符串拼接" (String Concatenation).

The "选择应用" section has a dropdown menu set to "字符串拼接" (String Concatenation). Below this, there is a table with three rows, each representing an input variable and its corresponding value:

变量名	变量值
String1	str. create_base - url
String2	str. create_table - table_id
String3	str. create_table - default_v...

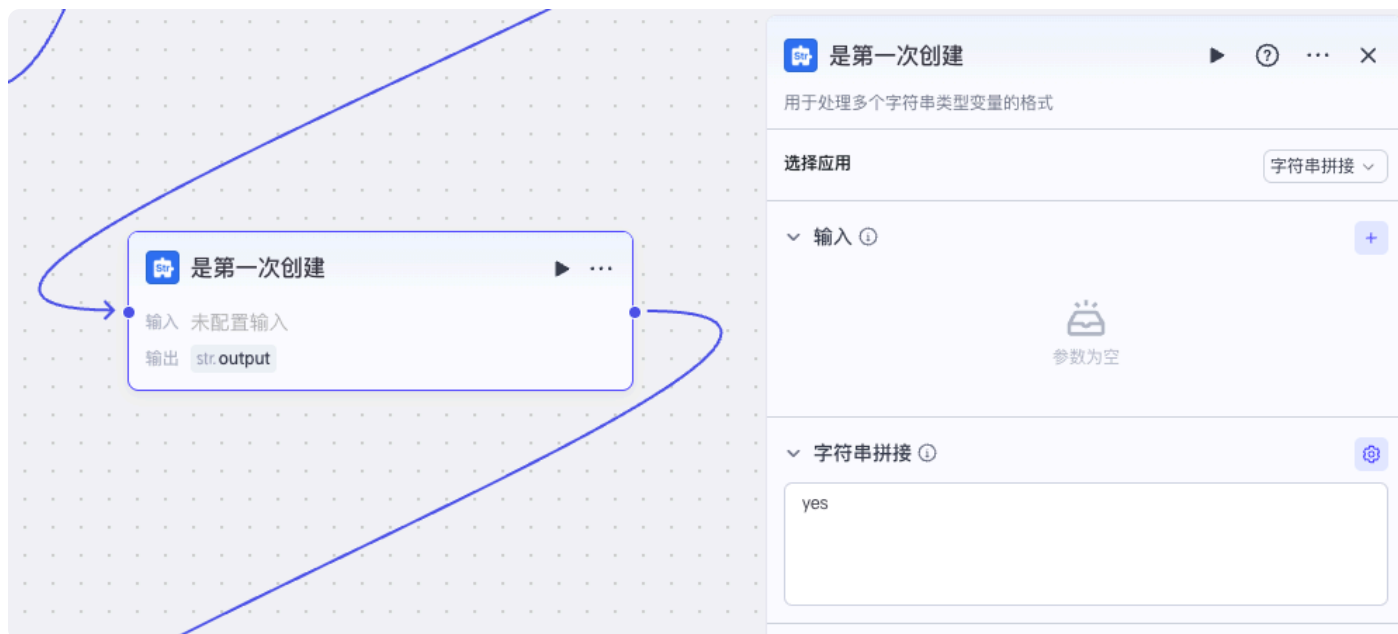
The "字符串拼接" section has a text area containing the following code:

```

{{String1}}?table={{String2}}&view={{String3}}

```

因为这个分支是第一次使用收藏夹助手，我需要将这信息也输出给结束节点，它对后续工作流有用，所以我用一个文本处理节点，生成一个常量 yes。

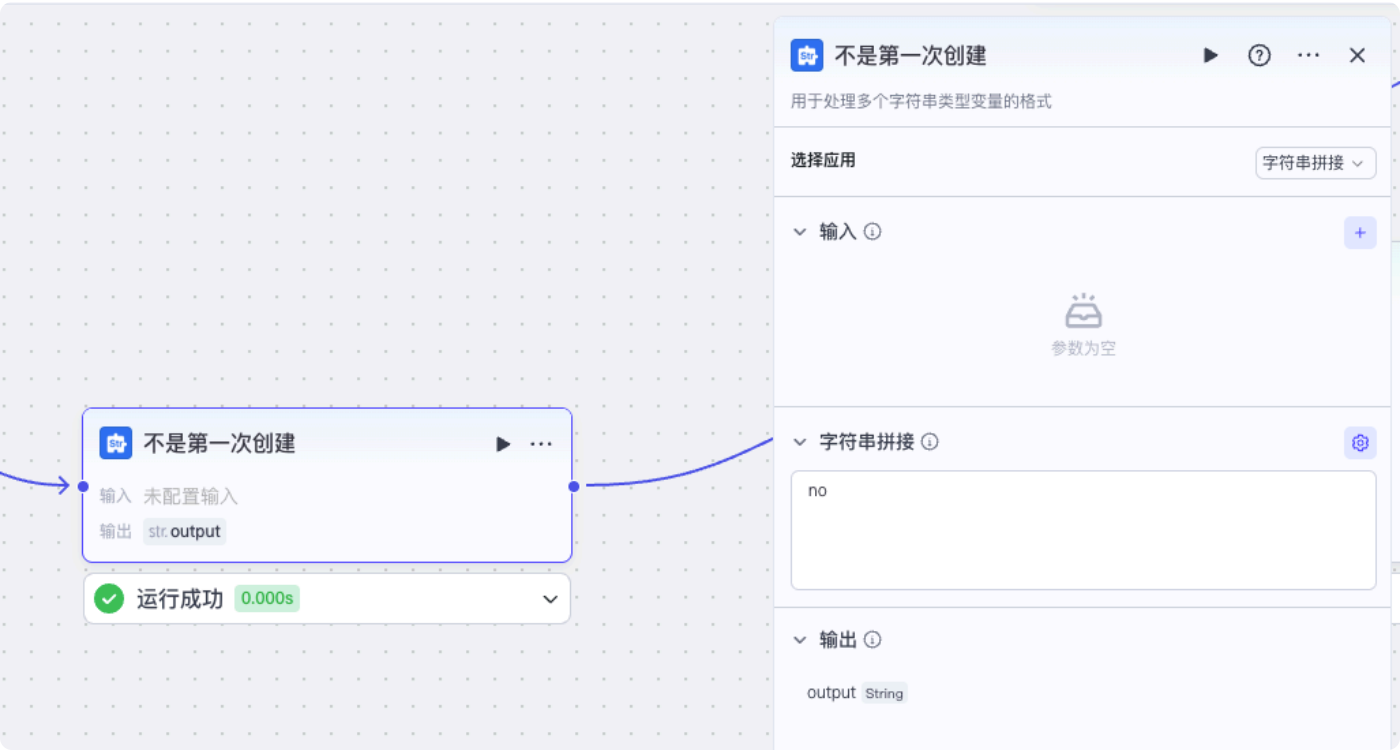


至此这个分支就完成了。

然后是 table\_url 不为空的分支。这时我们先通过 site\_url 查一下记录在数据库表里是否存在。



而走这个分支表示不是第一次使用收藏助手，我们同样用文本处理节点记一个常量 no。



接着我们将两个分支聚合起来。

The image shows a workflow editor interface. On the left, a node titled '文档URL和record\_id' is highlighted with a blue box. It contains three groups:

- Group1: str.String, str.output (str.table\_url)
- Group2: str.String, str.record\_id
- Group3: str.String, str.output (str.output)

Below the node, a status bar indicates '运行成功' (Run Success) with a green checkmark and '0.001s'.

On the right, the configuration panel for the '文档URL和record\_id' node is shown. It has a title bar with a question mark, three dots, and a close button. Below the title bar, it says '对多个分支的输出进行聚合处理' (Aggregate processing for multiple branch outputs).

The '聚合策略' (Aggregation Strategy) section shows a dropdown menu set to '返回每个分组中第一个非空的值' (Return the first non-empty value in each group).

The 'Group1 String' section shows three input fields:

- str. 我的收藏表 - output
- str. 查询数据 - table\_url
- str. 输入或引用参数值

The 'Group2 String' section shows two input fields:

- str. 查询数据\_1 - record\_id
- str. 输入或引用参数值

The 'Group3 String' section shows three input fields:

- str. 是第一次创建 - output
- str. 不是第一次创建 - output
- str. 输入或引用参数值

Below these sections is a button labeled '+ 新增分组' (Add New Group).

The '输出' (Output) section shows three output fields:

- Group1 String
- Group2 String
- Group3 String

这里 Group1 是多维表格的 URL；Group2 是 record\_id，如有，则表示网址已经收藏过；Group3 是一个标记，表示是否第一次使用收藏助手，它的值可能是 yes 或 no。

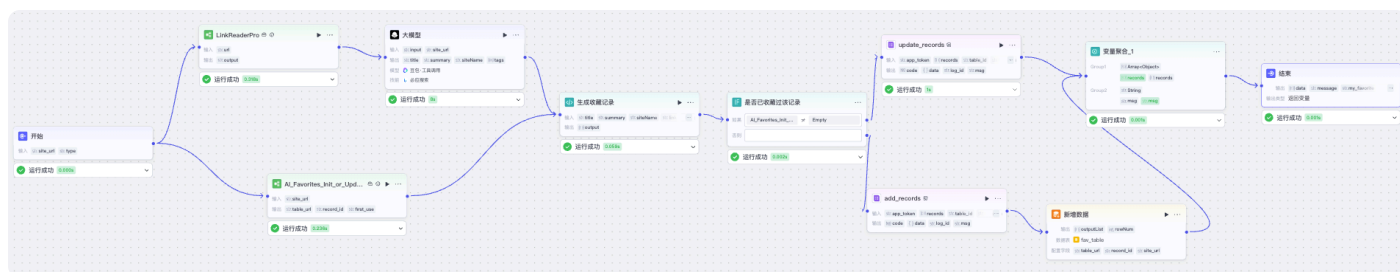
最终我们将数据设置给结束节点，测试后将 workflow 发布即可。





## 收藏网址主流程

接下来就是收藏网址的主体流程了：



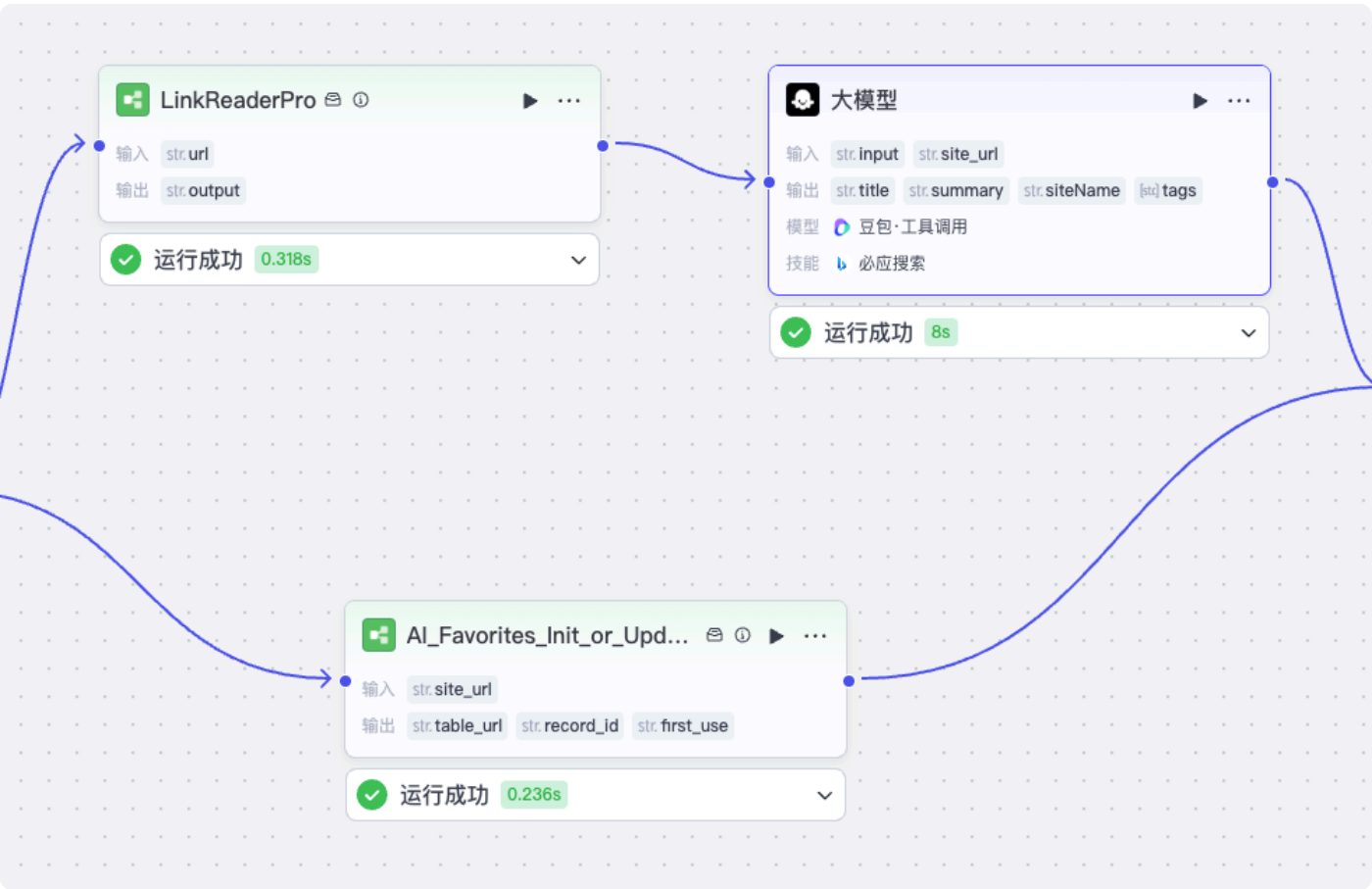
我们一步步来看。

首先开始节点接受两个参数，site\_url 必填，type 非必填，默认值为“网页”。



type 参数我们暂时用不着，但是这个保留，为了后续转写和笔记功能可以复用。

接着我们并行两个流程，一个是读取网页链接并用大模型提取内容，一个是初始化或查找是否已收藏该网页。



注意这个提取内容的大模型节点，我们直接用默认的豆包：



系统提示词和用户提示词分别如下：

## 系统提示词

复制代码

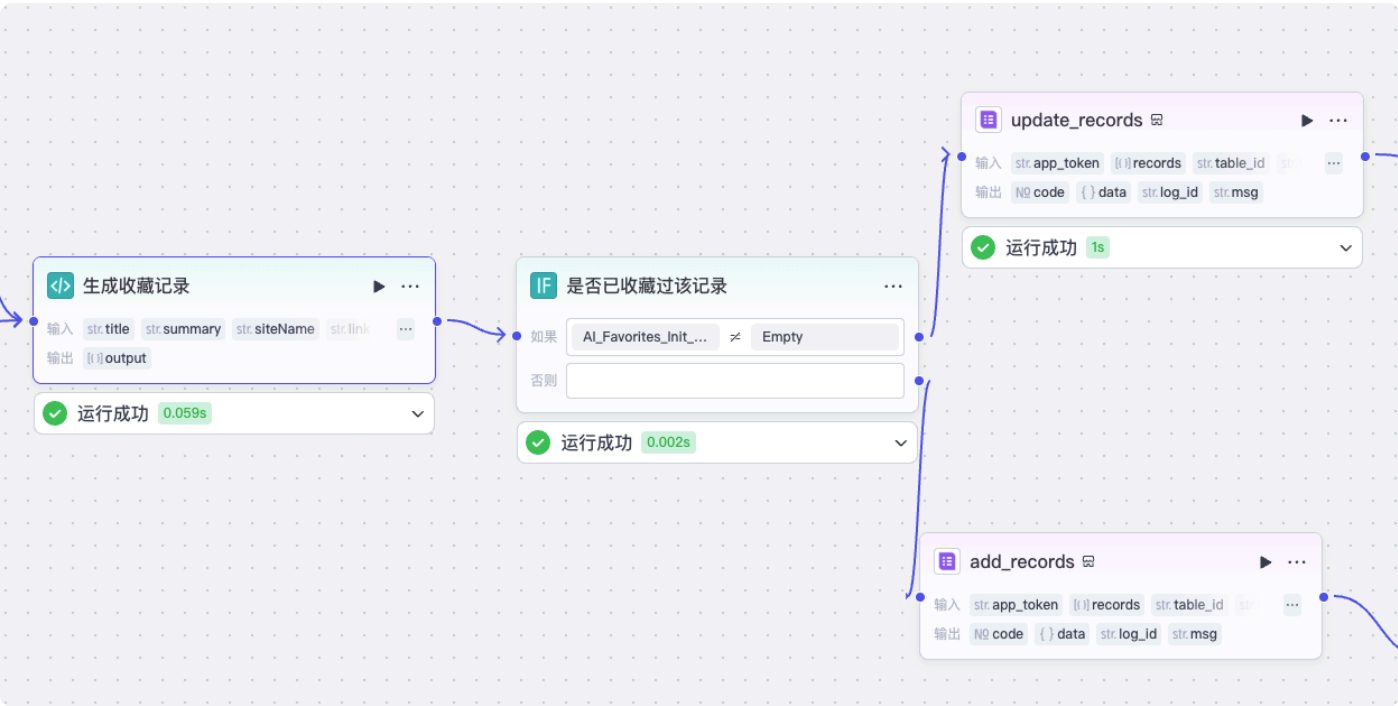
```
1 # 任务
2 根据用户输入,生成对应信息
3
4
5 # 输出要求
6 title: 网页标题 (如果有原标题,直接使用原标题;若找不到原标题,则根据关键信息,生成一个精确的标题)
7 summary: 仔细阅读全文内容,捕捉内容主题、关键信息、阅读价值,生成一段简洁而全面的摘要;并指出适合的读者群体
8 tags: 网站标签,类型为字符串数组
9 siteName: 网站平台名称,请根据site_url中的二级域名判断,如果不知道则直接填site_url中的二级域
```

用户提示词：

复制代码

```
1 # 内容
2 {{input}}
3
4 # 来源
5 {{site_url}}
```

内容提取完成之后，我们用一个代码节点生成收藏记录，并且根据 AI\_Faviors\_Init\_or\_Updated 工作流中返回的 record\_id 是否存在来决定是新增还是更新记录。



生成收藏记录的代码节点代码逻辑如下：

复制代码

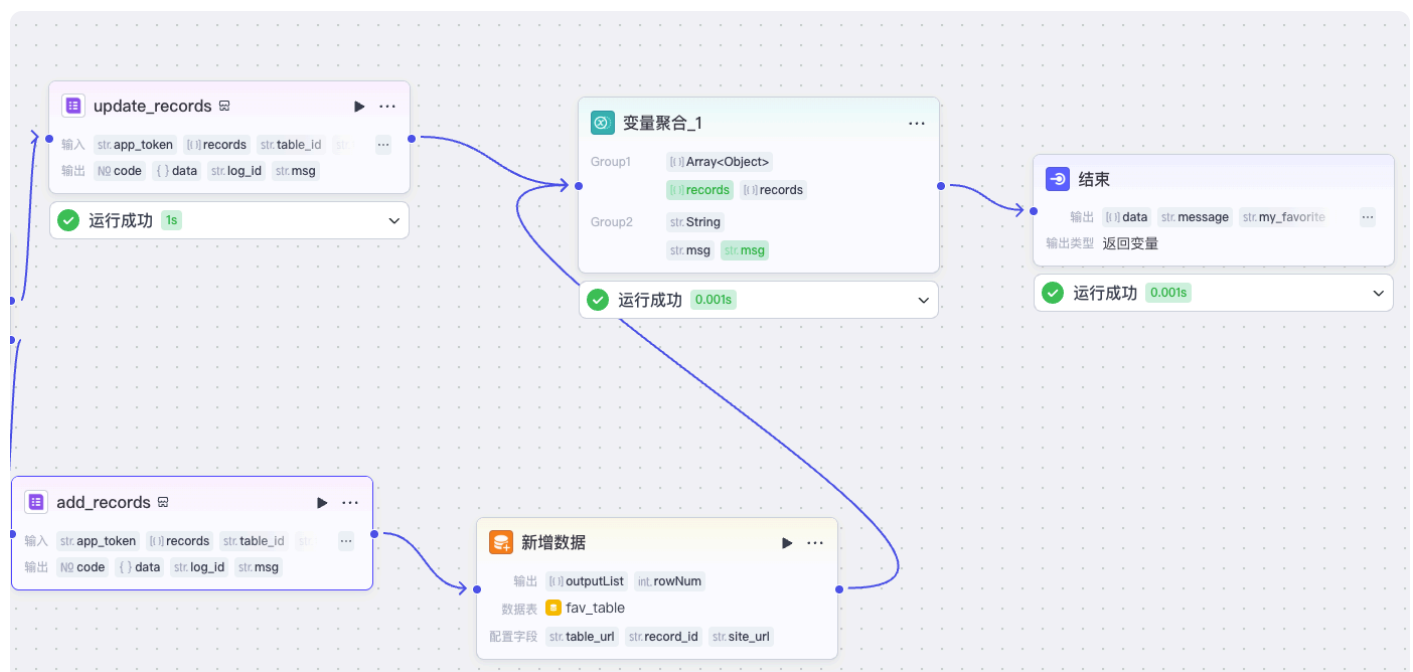
```
1 async function main({ params }: Args): Promise<Output> {
2     const {title, summary, siteName, link, tags, record_id, status} = params;
3
4     const fields = {
5         "内容": {
6             link,
```

```

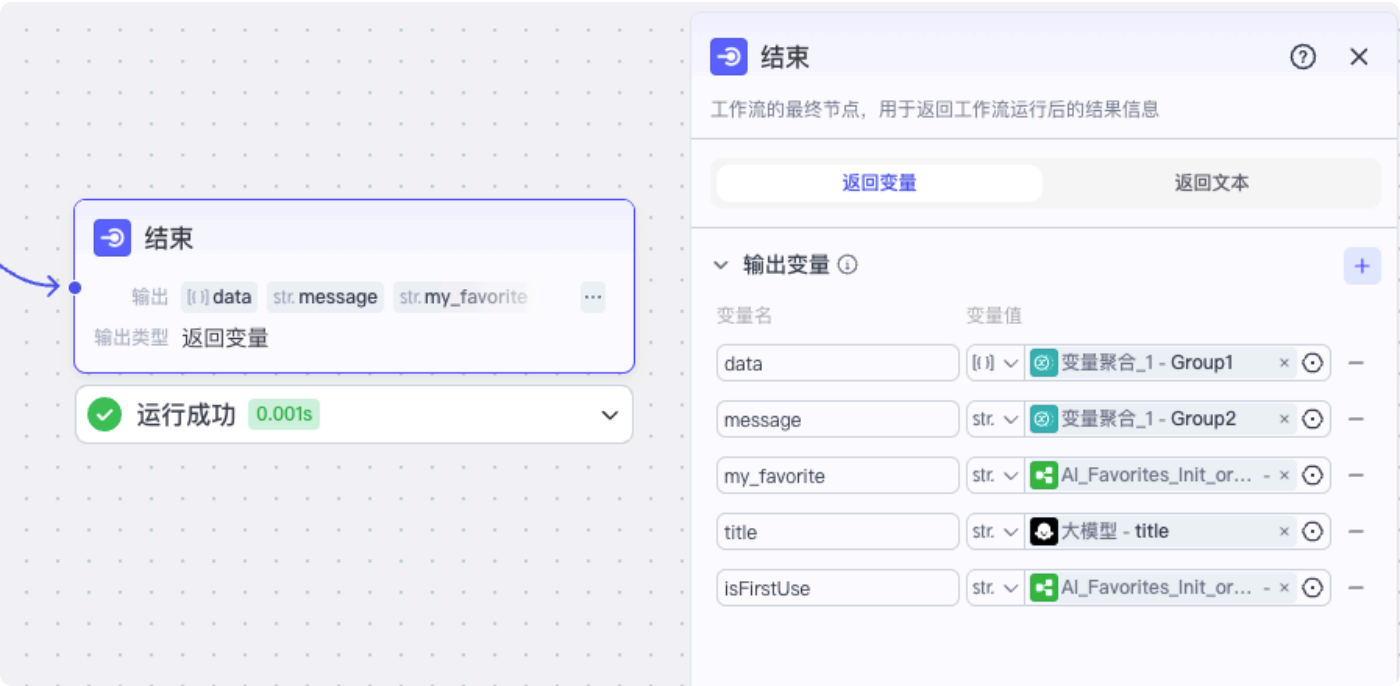
7         text: title,
8     },
9     "摘要": summary,
10    "平台": siteName,
11    "标签": tags,
12    "收藏时间": Date.now(),
13    "状态": status || '网页',
14 };
15
16 const output0: any = {fields: JSON.stringify(fields)};
17 if(record_id) output0.record_id = record_id;
18
19 // 构建输出对象
20 const ret = {
21     "output": [
22         output0,
23     ]
24 };
25
26 return ret;
27 }

```

我们根据 record\_id 是否存在的不同情况，使用飞书多维表格官方的 update\_records 插件或 add\_records 插件来更新收藏记录数据到多维表格，如果是 add\_records，那么我们还要将新增数据写入数据库，这个可以通过新增数据节点，配置数据库表 fav\_table 和数据字段来实现。



最后我们将结果通过结束节点输出，就完成了智能收藏助手收藏网页的核心 workflow。



## 要点总结

到此为止，我们的智能收藏主题流程就完成了，它比较复杂，可能是我们课程到目前为止遇到的最复杂的工作流。

但是在这里，很关键的一点是，我们采用了**自顶向下的思路来设计和拆解流程**，你会发现我把它拆成了三个流程，其中**链接读取子流程**、**初始化或查找收藏内容子流程**成为两个独立的工作流，这样方便我们独立开发和测试，然后我们再通过收藏网页主流程将它们整合在一起。

这是一种非常重要的 AI 工作流设计思想，希望大家能够通过学习这节课牢牢掌握。

智能收藏助手的内容还未结束，我们才实现了基础的网页收藏功能。下一节课，我们将继续讨论，在这个网页收藏工作流的基础上，逐步实现其他功能，最终实现这个完整应用。

## 课后练习

我们目前只是实现了收藏网页核心 workflow，并没有将它和任何智能体绑定，在后续实现其他功能后，我会最后创建智能体，实现完整的助手应用。但是你可以在这一节课后自己提前创建一个智能体，先试用这个收藏网页的核心功能效果。将你的创建过程和体验分享到评论区吧。

AI智能总结

- 1. 本文介绍了如何利用AI智能体来管理笔记和收集信息，强调了AI在个人能力提升方面的机会。
- 2. 文章详细介绍了一个基于AI的智能收藏助手的功能，包括收藏网址、转写英文资料、记录笔记以及管理资料等功能。
- 3. 作者解释了实现收藏网址功能的工作流和初始化或查找已收藏内容子流程的具体逻辑。
- 4. 文章提到了收藏网址的主体流程，包括读取网页链接并提取内容以及初始化或查找是否已收藏该网页等步骤。
- 5. 智能收藏助手的内容还未结束，下一节课将继续讨论，在网页收藏工作流的基础上，逐步实现其他功能，最终实现完整的应用。
- 6. 采用了自顶向下的思路来设计和拆解流程，将其拆成了三个流程，方便独立开发和测试，然后再通过收藏网页主流程将它们整合在一起。
- 7. 重要的AI工作流设计思想是自顶向下，希望读者能够通过学习这节课牢牢掌握。
- 8. 课后练习建议提前创建一个智能体，试用收藏网页的核心功能效果，并分享创建过程和体验到评论区。
- 9. 文章提供了关于如何利用AI智能体管理笔记和收集信息的详细指南，为读者提供了实用的信息管理方法。
- 10. 文章强调了开发者可以利用技术为自己的学习和成长打造最适合的效率工具，突出了AI技术在个人能力提升方面的潜力。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

全部留言 (1)

最新 精选



Chin 是我啊

2025-06-16 来自北京

第二个workflow前后对不上，有点乱 建议直接把链接贴出来吧



