

20 | 如何实现插图和听书功能

月影 · 跟月影学前端智能体开发



你好，我是月影。

回顾上一节课，我们实现了波波熊学伴的主体流程，这样从用户输入好奇心问题，到最后生成文字内容的全过程就已经完整了。

但是，因为波波熊学伴是给孩子使用的产品，而对于低年龄的孩子来说，太多的文字会让他们觉得内容过于深奥，从而影响他们的学习效果。

因此我们希望波波熊学伴不只是文字，而且能够有一些图片和语音，甚至在将来，还能够在页面上多一些互动。

那我们先来看图片和语音功能具体如何设计和实现。

实现独立的“插图”功能


如果你仔细学习了前面的课程，一定会发现我们其实为段落生成了插图提示词，但是我们没有在工作流中将图片直接生成出来，这是为什么呢？

其实这是产品为了控制成本的权衡，因为相较于文字来说，生成图片的成本较高，目前大概每张图 1-2 角钱。如果我们给每一个问题都生成了好几张图片，那么回答一个问题的成本就要是现在的好几倍，所以我们做了一个权衡，当用户主动要求生成图片的时候，才让 AI 为用户生成图片。

那么我们现在来看如何实现这一功能。

我们还是用 Trae 打开 Bearbobo Discovery 项目。


首先，我们要在 server.ts 文件里添加一个独立的生成插画的接口，因为我们在主工作流中已经用到了，`generate-image.ts` 模块来生成封面图，在前面的课程里我们已经完成了它的封装，所以 server.ts 里，实现的接口就非常简单，代码如下：

 复制代码

```
1 app.post('/gen-image', async (req, res) => {
2   const { prompt } = req.body;
3
4   const { url } = await generateImage(prompt);
5
6   res.json({
7     url
8   });
9 });
```

这里我们就直接复用主工作流里用到的 `generateImage` 方法就可以了。

接着我们调整前端逻辑。首先我们修改 `/src/components/BookDetails.vue`：

 复制代码

```
1 <script setup lang="ts">
2 ...
3 import { ref, type PropType, type Ref } from 'vue';
```

```

4  ...
5  const pictures: Ref<string[]> = ref([]);
6
7  const addPic = async (index: number, image_prompt: string) => {
8      const picture = pictures.value[index];
9      if (picture) return;
10     pictures.value[index] = 'https://res.bearbobo.com/resource/upload/e80ED0Jz/lc
11     const res = await fetch('/api/gen-image', {
12         method: 'POST',
13         headers: {
14             'Content-Type': 'application/json',
15         },
16         body: JSON.stringify({
17             prompt: image_prompt,
18         }),
19     });
20     const data = await res.json();
21     pictures.value[index] = data.url;
22 }
23 </script>
24
25 <template>
26 ...
27     <div class="article">
28         <div v-for="(topic, index) in topics" :key="index"
29             :class="{ 'topic': true, 'odd': index % 2 === 0, 'even': index %
30             <div class="topic-title">
31                 <h3>{{ topic.topic }}</h3>
32                 <div v-if="topic.image_prompt" class="btn" @click="addPic(ind
33                 </div>
34                 
35                 <p class="article_paragraph" v-html="marked.parse(topic.article_p
36                 <p class="post-reading-question">{{ topic.post_reading_question }
37             </div>
38         </div>
39     ...
40 </template>
41
42 <style scoped>
43 ...
44 .topic-title {
45     display: flex;
46     flex-direction: row;
47     margin-top: 40px;
48 }
49
50 .topic:first-child .topic-title {
51     margin: 0;
52 }

```

```

53
54 .topic-title h3 {
55     margin: 0;
56 }
57
58 .topic-title .btn {
59     margin-right: 20px;
60     cursor: pointer;
61 }
62
63 .topic-title .btn:first-of-type {
64     margin-left: auto;
65 }
66
67 .topic-title .btn::before {
68     content: "< ";
69 }
70
71 .topic-title .btn::after {
72     content: " >";
73 }
74
75 /* 奇数段落：图左文右 */
76 .topic.odd img {
77     float: left;
78     margin-right: 20px;
79 }
80
81 /* 偶数段落：图右文左 */
82 .topic.even img {
83     float: right;
84     margin-left: 20px;
85 }
86
87 /* 图片样式 */
88 .topic img {
89     width: 200px;
90     height: auto;
91     border-radius: 8px;
92     margin-top: 20px;
93 }
94 </style>

```

我们调整了一些组件的展现和功能逻辑。在结构上，在每个段落的标题后边，我们加上了一个配图的按钮；在内容结构里，如果有图片 URL，我们就会展示一张图片。

注意一个前端细节，为了让文章更加生动，我们配图的时候做了一个设计，当图片在奇数段落时，我们让图片被文字环绕在左侧；否则，当图片在偶数段落时，我们让图片被文字环绕在右侧。这一功能是通过给容器元素分别添加 `odd` 和 `even` class，然后通过控制 CSS 样式来实现的。

最后，我们在配图按钮被点击的时候，调用 `addPic` 异步方法。该方法会先在段落中展示出一张 loading 占位图，然后用我们已经拿到的 `image_prompt` 数据调用 `/api/gen-image`，拿到接口返回的图片 URL，再通过 `img` 标签显示出来。

这样我们就完成了文章配图的功能，注意这里我们对业务逻辑进行了简化。实际上完整的业务逻辑里，我们需要将生成的图片和文章绑定，将两者一同保存到数据库里。这样下次用户打开同样的文章，就可以把之前生成的图片显示出来。但数据入库并不是我们这门课程的重点，故而被我省略掉了。

这样我们就完成了文章配图功能，它的实际效果如下图：



狮子和老虎谁更厉害？

小朋友们，你们知道在森林里有许多动物，其中有一些特别厉害，大家都喜欢谈论它们。比如说狮子和老虎，它们都是很厉害的动物。那狮子和老虎，你们觉得哪个更强壮呢？让我们一起了解一下它们的特点，看看谁能成为森林中最强壮的动物吧！

狮子和老虎的体型有什么不同？

< 配图 >

在动物园里，我们经常能看到两种非常威武的大型猫科动物——狮子和老虎。想象一下，你在森林中散步，突然看到一只狮子，它会是什么样子的呢？狮子体型庞大，肩高(就是从脚底到肩膀的高度)可达1米左右，肩宽胸厚，就像一个小山丘一样壮实。狮子的头圆圆的，耳朵短小，还有一张大嘴巴，看起来很威武。但是，狮子的肚子却很柔软，像一块海绵一样。而狮子的鬃毛，就像围裙一样环绕在脖子周围，有的狮子鬃毛浓密，有的则比较稀疏。这种鬃毛不仅让狮子看起来更加威武，而且还有很多作用哦，比如保暖、保护脖子等。

现在让我们再来看看老虎，它们和狮子比起来有什么不一样呢？老虎的体型同样很大，但比狮子更加修长、灵活。老虎的体长可以达到3米多，在猫科动物中是体长最长的。老虎的身形看起来像一条大大的毛毛虫，不过这个毛毛虫可不好惹。老虎的尾巴也很长，占到体长的一半以

实现“听书”功能

除了插图以外，波波熊学伴还提供了听书的功能，但该功能只针对付费会员。

之所以这么设计，也是经过权衡的。与插图不一样，大段的文字转语音价格虽然也很贵，但是当用户想要听内容的时候，再去实时转换，这样的效果也不是很好，因为毕竟图片还可以通过占位符，让用户先阅读文字，等待图片的生成完成。

而当用户立即要听语音时，如果再去让用户等待，就有点消耗用户的耐心了，因此最终我们还是把语音转换流程添加到了主工作流中，只是把这个功能只开放给付费订阅的用户，借此控制成本。

那么我们现在看一下，添加语音功能到主工作流具体要怎么做。

口语化的听书


波波熊学伴的听书，并不是直接把文章里的内容原封不动地念出来。因为如果那样做，就变成朗读文章了，不够口语化，读起来也就不够有趣。

所以波波熊学伴做了细节上的改进，我们不是直接将书面语的文章内容转成语音，而是让大模型做了口语化改写，再将改写的内容转为语音。

我们来看一下具体怎么实现。

首先我们需要文字转语音的模块，这个在我们前面做拍照记单词的课程里已经创建过，我们直接拿来使用就行，将它放在 `lib/services/audio.ts` 文件中。

然后我们需要一个改写书面语内容到口语的 bot，先在 `/lib/prompts/podcast.tpl.ts` 文件中创建它的提示词：

 复制代码


```
1 export default `# Role and Goals
2 你是一个风趣幽默、擅长以生动有趣且简单清晰的语言进行科普教学的主播，你需要参考我发给你的文本段落，
3  - 确保使用简体中文输出纯文本，不要使用任何表情符号或Emoji
4  - 不要在开头和学生打招呼，避免使用<AvoidKeywords>中的词语
5  - 避免使用任何引导性的短语，直接陈述事实或给出答案
6  - 保持与原文一致的叙事风格，保留原文中的情感表达和细腻描写
7  - 在每个情境的描述中加入更多的细节，让学生能更好地想象和理解
8  - 在保持易懂的基础上，补充加入一些相关的科学知识解释，帮助学生理解原文中相关的科学原理
9
```

```

10 ## AvoidKeywords
11    ['魔法', '超级英雄', '想象一下', '你知道吗']
12
13 ## Student Information:
14 - gender: {{gender}}
15 - age: {{age}}
16 - student location: 中国
17
18 ## Study Style
19 The students' learning style preferences
20 - Communication-Style: Simple and Clear
21 - Tone-Style: Interesting and Vivid
22 - Reasoning-Framework: Intuitive
23 - Language: 简体中文`;

```

接着我们改写 server.ts:

 复制代码

```

1  ...
2  import podcastTpl from './lib/prompts/podcast.tpl.ts';
3  import { generateAudio } from './lib/service/audio.ts';
4  ...
5
6  app.get('/generate', async (req, res) => {
7    ...
8    // 文章生成
9    bot.addListener('inference-done', (content) => {
10      const { topics } = JSON.parse(content);
11      // console.log(topics);
12      for (let i = 0; i < topics.length; i++) {
13        const topic = topics[i];
14
15        const bot = ling.createBot(`topics/${i}`);
16        bot.addPrompt(articleTpl, userConfig);
17        bot.addFilter({
18          article_paragraph: true,
19          image_prompt: true,
20        });
21        bot.addListener('string-response', ({ uri, delta }) => {
22          if (uri.endsWith('article_paragraph')) {
23            const podcastBot = ling.createBot('', undefined, {
24              quiet: true,
25              response_format: { type: 'text' }
26            });
27            podcastBot.addPrompt(podcastTpl, userConfig);

```




```

28         podcastBot.chat(delta);
29
30         podcastBot.addListener('inference-done', (content) => {
31             // console.log(content);
32             ling.handleTask(async () => {
33                 const audioData = await generateAudio(content);
34                 const tmpId = Math.random().toString(36).substring(7)
35                 audioBuffers[tmpId] = Buffer.from(audioData, 'base64')
36                 // console.log('create audio', `/api/audio?id=${tmpId}`)
37                 ling.sendEvent({ uri: `topics/${i}/audio`, delta: `/a
38             });
39         });
40     }
41 });
42 bot.addListener('inference-done', (content) => {
43     console.log(JSON.parse(content));
44 });
45 bot.chat(JSON.stringify(topic));
46 }
47 });
48 ...
49 }

```

我们在文章生成的主流程中，创建博客 bot，然后将文章改写成口语风格，并转换为语音。这个步骤我们在前面的课程中已经学习过，具体转换逻辑如下：


 复制代码

```

1  podcastBot.addListener('inference-done', (content) => {
2      // console.log(content);
3      ling.handleTask(async () => {
4          const audioData = await generateAudio(content);
5          const tmpId = Math.random().toString(36).substring(7);
6          audioBuffers[tmpId] = Buffer.from(audioData, 'base64');
7          ling.sendEvent({ uri: `topics/${i}/audio`, delta: `/api/audio?id=${tmpI
8      });
9  });
10

```

与之前拍照记单词的课程里一样，我们还需要一个接口以文件的方式请求语音数据，避免将语音的二进制或者 base64 内容直接放到流式接口中，造成接口堵塞。


 复制代码

```
1 const audioBuffers: Record<string, Buffer> = {};  
2  
3 app.get('/audio', (req, res) => {  
4   const id = req.query.id as string;  
5   const audioData = audioBuffers[id];  
6   if (!audioData) {  
7     res.status(404).send('Audio not found');  
8     return;  
9   }  
10  res.setHeader('Content-Type', 'audio/ogg');  
11  res.send(audioData);  
12 });
```

这样我们就改写好了 `server.ts` 的逻辑，接下来要实现前端交互。


我们还是修改 `BookDetails.vue` 组件。

首先我们在代码中添加 `Topic` 对象的 `audio` 属性，然后增加 `playAudio` 方法。由于我们在 `server.ts` 中，已经通过 `ling.sendEvent` 将 `audio` 的 url 通过 `topics/${i}/audio` 发送给前端，所以我们的组件直接可以拿到这个数据。

 复制代码

```
1 <script setup lang="ts">  
2 ...  
3 interface Topic {  
4   topic: string,  
5   post_reading_question: string,  
6   article_paragraph?: string,  
7   image_prompt?: string,  
8   audio?: string,  
9 }  
10  
11 ...  
12 const playAudio = (audioUrl: string) => {  
13   const audio = new Audio(audioUrl);  
14   audio.play();  
15 }  
16 </script>
```

然后我们在 `.topic-title` 元素中，添加一个新的按钮：

 复制代码

```
1 <template>
2 ...
3   <div class="topic-title">
4     <h3>{{ topic.topic }}</h3>
5     <div v-if="topic.image_prompt" class="btn" @click="addPic(index, topic.ir
6     <div v-if="topic.audio" class="btn" @click="playAudio(topic.audio)">听书<,
7   </div>
8 </template>
```

当 `topic.audio` 数据返回的时候，这个听书按钮就会出现在页面上，在用户想要听书的时候，点击按钮直接播放对应的音频就可以了。

最终，文章内容转换语音效果如下：

 [audio.ogg](#)

要点总结

以上就是波波熊学伴插图和听书的整体功能了，在这里为了讲解方便，我做了一些简化，比如“听书”在真正的波波熊学伴产品里，还实现了一个播放器界面，在播放语音的同时显示文字内容。在课程中为了便于理解我省略掉了这部分，但并不影响核心技术实现。

在这节课中，最重要的还是进一步理解 **AI 大模型应用的异步 workflows 开发思路**。要完全掌握这些技巧，光听我讲是不够的，大家还是应该要多动手实践。

完整的代码在 [Github 仓库](#)里，你可以拉代码下来运行和自行修改研究。

课后练习

可能仔细学习的小伙伴会发现，我们的配图功能配置出来的图片形态相对比较单一，多样性和丰富性不够，或者说不是很生动。这是因为，我其实在提炼这个实战项目的时候，为了便于大

家理解技术内容，对一些固有产品逻辑的细节做了简化。

实际上我们这里用的配图提示词，在原本的产品里是用来生成奖励卡片的，它会在孩子回答问题之后生成。原本插画的提示词生成另有一套流程，但是它并不影响我们具体的技术实现，所以我就没有再重复这套流程，以免课程变得越来越复杂。

那么问题来了，如果你想要为波波熊学伴生成更加生动、内容丰富的插图，你会怎么改进呢？

请认真思考一下，将你的想法分享到评论区，等后面有空，我可能会单独写一篇“加餐”的内容来讲讲究竟如何让 AI 生成更加生动有趣的内容，到时候有机会详细展开，把这块的方法和思路再给同学们多说一说。

AI智能总结

1. 实现独立的“插图”功能，通过控制成本，只有当用户主动要求生成图片时，才让AI为用户生成图片。
2. 调整前端逻辑，实现文章配图功能，包括在每个段落的标题后加上配图按钮，根据图片URL展示图片，并通过CSS样式控制图片在奇数段落和偶数段落的位置。
3. 实现“听书”功能，将语音转换流程添加到主工作流中，但该功能只对付费订阅的用户开放，以控制成本。

© 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。

精选留言

由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。