**软件学院2032-2024学年度第二学期**

**《软件工程理论与实践课程设计》**

**实验报告**

|  |  |
| --- | --- |
| **班级：** | 软件工程2203班 |
| **项目名称：** | TransLink\_PW |
| **项目组组长：** |  |
| **项目组成员：** | 学号：U202210416  姓名：  学号：0002  姓名：李四 **李四手写签名确认** |
| **指导老师：** | 刘小峰 |

**目录**

[1. 项目概述 3](#_Toc168591146)

[2. 需求描述 3](#_Toc168591147)

[2.1 功能性需求 3](#_Toc168591148)

[2.2 非功能性需求 3](#_Toc168591149)

[3. 系统设计[深度学习部署类题目可以不写] 3](#_Toc168591150)

[3.1 架构设计 3](#_Toc168591151)

[3.2 界面原型设计 3](#_Toc168591152)

[3.3 详细设计 3](#_Toc168591153)

[4. 系统实现 3](#_Toc168591154)

[5. 系统界面展示 4](#_Toc168591155)

[6. 总结 4](#_Toc168591156)

## 项目概述

### 项目基本介绍

**UI界面：**

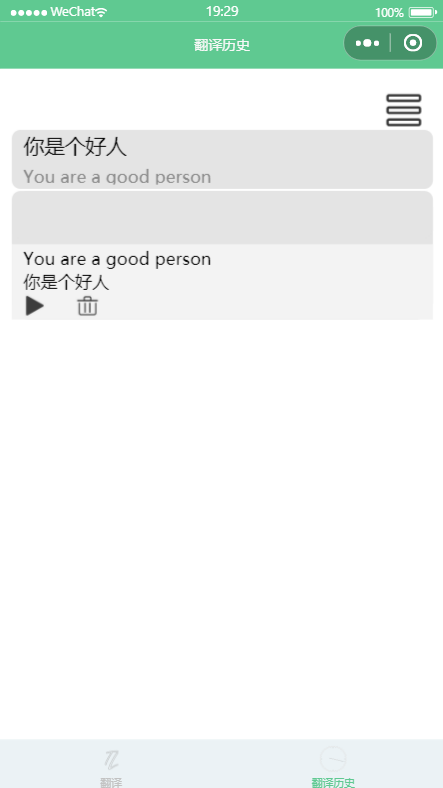
 

图1- 1UI界面

**基本介绍：**

TransLink\_PW是一款集多功能于一身的微信小程序，它不仅能够将输入文本快速准确地翻译成目标语言，还能够提供语音识别，图片识别快速地帮助用户获得输入文本。并且TransLink\_PW还提供了历史记录的功能，让用户能够轻松地管理翻译历史记录。

### Github仓库地址

[2042270217/TransLink\_PW (github.com)](https://github.com/2042270217/TransLink_PW)

### 人员基本分工

**潘隆财：**主要负责前端的开发和部分文档的撰写

1. 共同撰写文档：一起参与讨论于构思，完成需求分析，系统设计，界面原型设计等文档。
2. 原型设计：使用Mastergo设计小程序的初代界面原型。
3. 界面编写：完成了TransLink、History、ClipPicture页面的编写
4. 组件开发：为了更加方便地编写History界面，开发了singleHistory组件

## 需求描述

### 功能性需求

### 非功能性需求

## 系统设计

### 架构设计

### 界面原型设计

#### 翻译界面

翻译界面是小程序的入口，包含了小程序的主要功能，包括用户输入模块，语言选择模块，翻译输出模块以及翻译方式选择。

* 输入框：用户可以在这个框中手动输入需要翻译的文本。
* 语音播放器（输入框中）：用户点击按钮可以播放需要翻译的文本的语音。
* 复制按钮（输入框中）：用户点击按钮可以复制输入框的内容到剪切板
* 翻译按钮：用户点击这个按钮后，翻译框会显示翻译后的文本。
* 语言选择框：用于点击左边的按钮可以选择源语言，点击右边的按钮可以选择目标语言，点击中间的按钮可以交换源语言和目标语言。
* 翻译框：用户可以在这个框中看到翻译后的文本。
* 语音播放器（翻译框中）：用户点击按钮可以播放翻译后的文本的语音。
* 复制按钮（翻译框中）：用户点击按钮可以复制翻译框的内容到剪切板
* 麦克风按钮：用户点击按钮可以开始录制语音，再次点击结束录制
* 相机按钮：用户点击后可以选择拍照或者上传图片进行翻译，随后进入裁剪界面

#### 历史记录界面

历史记录界面可以查看以及删除翻译历史记录

* 多选按钮：用户点击后可以激活左侧的复选框
  + 全选按钮：用户点击后可以全选复选框
  + 删除按钮：用户点击后可以删除选择的历史记录
* 单个历史记录：用户点击后可以展开历史记录
  + 语音播放器：用户点击按钮可以播放翻译后的文本的语音。
  + 删除按钮：用户点击后可以删除该条历史记录

#### 裁剪界面

裁剪界面可以裁剪用户上传的图片，小程序会识别图片的文字进行翻译

* 裁剪框：用户拖动裁剪框可以设置裁剪的范围，单击裁剪框可以预览裁剪后的图片
* 裁剪按钮：用户点击后小程序自动裁剪图片，并且跳转到翻译界面

### 详细设计

## 系统实现

### components/

#### components/image-cropper

image-cropper是在微信开发者社区中寻找的插件，在项目中实现了裁剪图片的主要功能

#### components/singleHistory

singleHistory组件定义了历史记录界面中单个历史记录的呈现形式，主要实现的功能有：

* 记录输入文本与输出文本信息，以及管理历史记录所需的id
* 能够实现单击转换展开和折叠的样式

singleHistory.js

1. Component({
2. */\*\**
3. \* 组件的属性列表
4. \*/
5. properties: {
6. currentText:{
7. type:String,
8. value:'历史'
9. },
10. outputText:{
11. type:String,
12. value:'History'
13. },
14. index:{
15. type:Number,
16. value:0
17. }
18. },
19. */\*\**
20. \* 组件的初始数据
21. \*/
22. data: {
23. shouldUnfold:false
24. },
25. */\*\**
26. \* 组件的方法列表
27. \*/
28. methods: {
29. checkDel(){
30. this.triggerEvent('del',this.properties.index)
31. },
32. unfold(){
33. this.setData({
34. shouldUnfold:!this.data.shouldUnfold
35. })
36. },
37. playCurrentVoice(){
38. console.log("playOutputVoice~" + this.data.outputText);
39. const generateVoice = require('../../utils/generateVoice.js');
40. generateVoice.GenerateVoice(this.data.outputText);
41. }
42. }
43. })

singleHistory.wxml

1. <block wx:if="{{shouldUnfold}}">
2. <view class="singleHistory\_big">
3. <view class="top" bind:tap="unfold">
4. </view>
5. <view class="mainBody">
6. <view class="textBody">
7. <text class="singleCurrentHistory">{{currentText}} </text>
8. <text class="singleOutputHistory">{{outputText}} </text>
9. </view>
10. <view class="buttonBody">
11. <image class="button" src="../../asset/pic/bf.png" mode="scaleToFill" bind:tap="playCurrentVoice" />
12. <image class="button" src="../../asset/pic/sc.png" mode="scaleToFill" bind:tap="checkDel" />
13. </view>
14. </view>
15. </view>
16. </block>
17. <block wx:else>
18. <view class="singleHistory\_normal" bind:tap="unfold">
19. <view class="textBody\_normal">
20. <text class="singleCurrentHistory\_normal" max-lines="{{1}}" overflow="clip">{{currentText}} </text>
21. <text class="singleOutputHistory\_normal" max-lines="{{1}}" overflow="clip">{{outputText}} </text>
22. </view>
23. </view>
24. </block>

singleHistory.scss：

1. .singleHistory\_normal {
2. height: 100rpx;
3. background-color: #e4e4e4;
4. margin-right: 20rpx;
5. margin-left: 20rpx;
6. margin-top: 5rpx;
7. border-radius: 15rpx;
8. }
9. .singleCurrentHistory\_normal {
10. height: 55rpx;
11. width: 650rpx;
12. margin-top: 5rpx;
13. overflow: hidden;
14. white-space: nowrap; *// 处理元素内的空白：不换行*
15. text-overflow: ellipsis; *//文本溢出处理方式：显示省略号*
16. font-size: larger;
17. }
18. .singleOutputHistory\_normal {
19. height: 35rpx;
20. width: 650rpx;
21. margin-bottom: 5rpx;
22. color: rgb(139, 139, 139);
23. font-size: medium;
24. overflow: hidden;
25. white-space: nowrap; *// 处理元素内的空白：不换行*
26. text-overflow: ellipsis; *//文本溢出处理方式：显示省略号*
27. }
28. .textBody\_normal{
29. display: flex;
30. flex-direction: column;
31. margin-top: 5rpx;
32. margin-bottom: 5rpx;
33. margin-right: 20rpx;
34. margin-left: 20rpx;
35. align-items: flex-start;
36. justify-content: center;
37. }
38. .singleHistory\_big {
39. max-height: 450rpx;
40. display: flex;
41. flex-direction: column;
42. background-color: #e4e4e4;
43. margin-right: 20rpx;
44. margin-left: 20rpx;
45. margin-top: 5rpx;
46. border-radius: 30rpx;
47. }
48. .top{
49. background-color: #e4e4e4;
50. border-radius:15rpx;
51. height: 80rpx;
52. }
53. .mainBody{
54. display: flex;
55. margin-top: 10rpx;
56. background-color: #f3f3f3;
57. flex-direction: column;
58. max-height: 350rpx;
59. }
60. .textBody{
61. display: flex;
62. flex:4;
63. flex-direction: column;
64. margin-top: 5rpx;
65. margin-bottom: 5rpx;
66. margin-right: 20rpx;
67. margin-left: 20rpx;
68. max-height: 300rpx;
69. }
70. .buttonBody{
71. flex:1;
72. margin-right: 20rpx;
73. margin-left: 20rpx;
74. margin-bottom: 5rpx;
75. display: flex;
76. gap: 50rpx;
77. max-height: 50rpx;
78. }
79. .button{
80. width: 40rpx;
81. height: 40rpx;
83. }

### pages/

#### pages/TransLink

翻译界面主要实现的功能有：

* 记录输入文本以及翻译后的输出文本
* 记录支持的语言以及输入与输出文本对应的语言
* 记录是否在录音状态
* 记录图片裁剪界面传递过来的图片URL

TransLink.js：

1. const app = getApp();
2. *//引入插件：微信同声传译*
3. const plugin = requirePlugin('WechatSI');
4. *//获取全局唯一的语音识别管理器recordRecoManager*
5. const manager = plugin.getRecordRecognitionManager();
6. Page({
7. */\*\**
8. \* 页面的初始数据
9. \*/
10. data: {
11. currentText: '',
12. outputText: '',
13. languageArr: [
14. "自动检测",
15. "中文",
16. "英语",
17. "西班牙语",
18. "法语",
19. "阿拉伯语",
20. "俄语",
21. "韩语",
22. "德语",
23. "日语",
24. "葡萄牙语",
25. "意大利语",
26. "荷兰语",
27. "泰语",
28. "希腊语",
29. "波兰语",
30. "捷克语",
31. "匈牙利语",
32. "瑞典语",
33. "芬兰语",
34. "保加利亚语",
35. "丹麦语",
36. "罗马尼亚语",
37. "斯洛文尼亚语",
38. "爱沙尼亚语",
39. "越南语",
40. "繁体中文",
41. "粤语",
42. "文言文",
43. ],
44. currentLanguage: 0,
45. targetLanguage: 1,
46. recordState: false,
47. imgURL: ''
48. },
49. */\*\**
50. \* 生命周期函数--监听页面加载
51. \*/
52. onLoad(options) {
53. this.initRecorder();
54. },
55. initRecorder() {
56. const that = this;
57. *// 正常开始录音识别时会调用此事件*
58. manager.onStart = function (res) {
59. console.log("成功开始录音识别")
60. }
61. *//识别结束事件*
62. manager.onStop = function (res) {
63. console.log("结束录音", res.result)
64. const re = res.result
65. that.setData({
66. currentText: re
67. })
68. }
69. manager.onRecognize = function (res) {
70. console.log(res.result)
71. }
72. },
73. */\*\**
74. \* 生命周期函数--监听页面显示
75. \*/
76. onShow() {
77. var app = getApp()
78. console.log(app.globalData.reservePhotoURL)
79. this.setData({
80. imgURL: app.globalData.reservePhotoURL
81. })
82. const pictureTranslate = require('../../utils/pictureTranslate.js');
83. const language = require('../../utils/language.js');
84. pictureTranslate.PictureTranslate(language.findLanguageCode(this.data.languageArr[this.data.currentLanguage]), language.findLanguageCode(this.data.languageArr[this.data.targetLanguage]), this,this.data.imgURL)
85. },
86. playCurrentVoice() {
87. console.log("playCurrentVoice~" + this.data.currentText)
88. const generateVoice = require('../../utils/generateVoice.js');
89. generateVoice.GenerateVoice(this.data.currentText);
90. },
91. confirmText(event) {
92. this.setData({
93. currentText: event.detail.value
94. })
95. },
96. translate() {
97. const generalTranslate = require('../../utils/generalTranslate.js');
98. const history = require('../../utils/history.js');
99. const language = require('../../utils/language.js');
100. generalTranslate.GeneralTranslate(this.data.currentText, language.findLanguageCode(this.data.languageArr[this.data.currentLanguage]), language.findLanguageCode(this.data.languageArr[this.data.targetLanguage]), this);
101. },
102. setCurrentLanguage(event) {
103. this.setData({
104. currentLanguage: event.detail.value
105. })
106. console.log(this.data.languageArr[this.data.currentLanguage])
107. },
108. setTargetLanguage(event) {
109. this.setData({
110. targetLanguage: event.detail.value
111. })
112. },
113. changeLanguage() {
114. const temp = this.data.currentLanguage
115. this.setData({
116. currentLanguage: this.data.targetLanguage
117. })
118. this.setData({
119. targetLanguage: temp
120. })
121. },
122. playOutputVoice() {
123. console.log("playOutputVoice~" + this.data.outputText);
124. const generateVoice = require('../../utils/generateVoice.js');
125. generateVoice.GenerateVoice(this.data.outputText);
126. },
127. acceptVoice() {
128. if (!this.data.recordState) {
129. manager.start()
130. this.setData({
131. recordState: true
132. })
133. } else {
134. manager.stop();
135. this.setData({
136. recordState: false
137. })
138. }
139. },
140. takePhoto() {
141. console.log("take a photo")
142. wx.chooseMedia({
143. count: 1,
144. mediaType: ['image'],
145. success(res) {
146. const photo = res.tempFiles[0].tempFilePath
147. wx.navigateTo({
148. url: `/pages/ClipPicture/ClipPicture?photoPath=${photo}`
149. })
150. }
151. })
152. },
153. setClipboardCur() {
154. wx.setClipboardData({
155. data: this.data.currentText
156. })
157. },
158. setClipboardOut() {
159. wx.setClipboardData({
160. data: this.data.outputText
161. })
162. }
163. })

TransLink.wxml：根据模块分为了输入文本框、语言选择框和输出文本框

1. *<!-- inputBox -->*
2. <view class="inputBoxBody">
3. <view class="inputBoxTextBody">
4. <textarea class="inputText" auto-focus confirm-type="done" placeholder="请输入文本" value="{{currentText}}" maxlength="100" bindinput="confirmText" />
5. </view>
6. <view class="inputBoxButton">
7. <view class="leftButton">
8. <image class="button" src="../../asset/pic/bf.png" mode="scaleToFill" bind:tap="playCurrentVoice" />
9. <image class="button" src="../../asset/pic/fz.png" mode="scaleToFill" bind:tap="setClipboardCur" />
10. </view>
11. <text class="text">{{currentText.length}}/100</text>
12. <view class="rightButton">
13. <image class="buttonBig" src="../../asset/pic/fy.png" mode="scaleToFill" bind:tap="translate" />
14. </view>
15. </view>
16. </view>
17. *<!-- selectLanguage -->*
18. <view class="selectBody">
19. <picker range="{{languageArr}}" value="{{currentLanguage}}" bindchange="setCurrentLanguage">
20. <text class="languageText">{{languageArr[currentLanguage]}}</text>
21. </picker>
22. <image class="selectButtonMid" src="../../asset/pic/qh.png" mode="scaleToFill" bind:tap="changeLanguage"></image>
23. <picker range="{{languageArr}}" value="{{targetLanguage}}" bindchange="setTargetLanguage">
24. <text class="languageText">{{languageArr[targetLanguage]}}</text>
25. </picker>
26. </view>
27. *<!-- outputBox -->*
28. <view class="outputBoxBody">
29. <view class="outputBoxTextBody">
30. <textarea class="outputText" value="{{outputText}}" auto-focus confirm-type="done" placeholder="翻译结果" disabled="true" />
31. </view>
32. <view class="inputBoxButton">
33. <view class="leftButton">
34. <image class="button" src="../../asset/pic/bf.png" mode="scaleToFill" bind:tap="playOutputVoice" />
35. <image class="button" src="../../asset/pic/fz.png" mode="scaleToFill" bind:tap="setClipboardOut" />
36. </view>
37. </view>
38. </view>
39. <view class="buttons">
40. <block wx:if="{{!recordState}}">
41. <image class="buttonItem" src="../../asset/pic/mkf.png" mode="scaleToFill" bind:tap="acceptVoice" />
42. </block>
43. <block wx:else>
44. <image class="buttonItem" src="../../asset/pic/bf.png" mode="scaleToFill" bind:tap="acceptVoice" />
45. </block>
46. <image class="buttonItem" src="../../asset/pic/xj.png" mode="scaleToFill" bind:tap="takePhoto" />
47. </view>

TransLink.scss：

1. .mainTexBody {
2. height: 200rpx;
3. display: flex;
4. flex-direction: row;
5. align-items: flex-end;
6. justify-content: space-around;
7. background-color: rgb(158, 214, 186, 0.3);
8. margin-right: 20rpx;
9. margin-left: 20rpx;
10. border-radius: 20rpx;
11. }
12. .mainTex {
13. height: 200rpx;
14. width: 200rpx;
15. }
16. .inputBoxBody {
17. display: flex;
18. height: 400rpx;
19. background-color: #e4e4e4;
20. margin-left: 20rpx;
21. margin-right: 20rpx;
22. margin-top: 10rpx;
23. margin-bottom: 10rpx;
24. border-radius: 30rpx;
25. flex-direction: column;
26. }
27. .inputBoxTextBody {
28. flex: 5;
29. margin-left: 20rpx;
30. margin-right: 20rpx;
31. margin-top: 10rpx;
32. margin-bottom: 10rpx;
33. flex-direction: row;
34. }
35. .inputBoxButton {
36. flex: 1;
37. display: flex;
38. margin-left: 20rpx;
39. margin-right: 20rpx;
40. margin-top: 10rpx;
41. margin-bottom: 10rpx;
42. align-items: center;
43. justify-content: space-around;
44. }
45. .leftButton {
46. display: flex;
47. flex: 3;
48. }
49. .rightButton {
50. display: flex;
51. flex: 1;
52. }
53. .button {
54. width: 50rpx;
55. height: 50rpx;
56. }
57. .buttonBig {
58. width: 70rpx;
59. height: 70rpx;
60. }
61. .inputText {
62. font-size: 35rpx;
63. }
64. .selectBody {
65. display: flex;
66. height: 80rpx;
67. background-color: #e4e4e4;
68. margin-left: 20rpx;
69. margin-right: 20rpx;
70. margin-top: 10rpx;
71. margin-bottom: 10rpx;
72. border-radius: 20rpx;
73. flex-direction: row;
74. align-items: center;
75. justify-content: space-around;
76. }
77. .languageText {
78. flex: 1;
79. font-size: 40rpx;
80. }
81. .selectButton {
82. width: 100rpx;
83. height: 50rpx;
84. }
85. .selectButtonMid{
86. width: 100rpx;
87. height: 50rpx;
88. position: absolute;
89. left:325rpx;
90. }
91. .outputBoxBody {
92. display: flex;
93. height: 400rpx;
94. background-color: #e4e4e4;
95. margin-left: 20rpx;
96. margin-right: 20rpx;
97. margin-top: 10rpx;
98. margin-bottom: 10rpx;
99. border-radius: 30rpx;
100. flex-direction: column;
101. }
102. .outputBoxTextBody {
103. flex: 5;
104. margin-left: 20rpx;
105. margin-right: 20rpx;
106. margin-top: 10rpx;
107. margin-bottom: 10rpx;
108. flex-direction: row;
109. }
110. .outputBoxButton {
111. flex: 1;
112. display: flex;
113. margin-left: 20rpx;
114. margin-right: 20rpx;
115. margin-top: 10rpx;
116. margin-bottom: 10rpx;
117. }
118. .leftButton {
119. display: flex;
120. flex: 3;
121. gap: 20rpx;
122. }
123. .text{
124. flex: 2;
125. font-size: large;
126. }
127. .rightButton {
128. display: flex;
129. flex-direction: column;
130. *// align-items: flex-end;*
131. *// justify-content:space-around;*
132. flex: 1;
133. }
134. .button {
135. width: 50rpx;
136. height: 50rpx;
137. }
138. .buttonBig {
139. width: 70rpx;
140. height: 70rpx;
141. }
142. .outputText {
143. font-size: 35rpx;
144. }
145. .buttons{
146. display: flex;
147. height: 200rpx;
148. flex-direction: row;
149. align-items:center;
150. justify-content: space-around;
151. }
152. .buttonItem{
153. height: 120rpx;
154. width: 120rpx;
155. }

#### pages/History

历史记录界面实现的功能主要有：

* 从app.globalData中获取需要渲染的历史记录信息
* 记录是否处在选择历史记录状态和多选状态
* 记录用户选择删除的历史记录的id

History.js：首先实现delSingle函数，再通过删除一个历史记录实现删除多个历史记录，从而管理历史记录数组

1. const app = getApp();
2. const history = require('../../utils/history.js')
3. Page({
4. */\*\**
5. \* 页面的初始数据
6. \*/
7. data: {
8. multiSelectState: false,
9. transHistory: [],
10. selectedHis: [],
11. allSelected: false
12. },
13. delSingle(index) {
14. var his = this.data.transHistory
15. history.removeHistory(his[index].key)
16. his.splice(index, 1)
17. for (var i = index; i < his.length; i++) {
18. his[i].id -= 1
19. }
20. this.setData({
21. transHistory: his
22. })
23. console.log(this.data.transHistory)
24. },
25. async checkDel(event) {
26. console.log("删除记录" + event.detail)
27. const res = await wx.showModal({
28. title: '提示',
29. content: '是否删除该历史记录',
30. })
31. if (res.confirm) {
32. this.delSingle(event.detail)
33. app.globalData.transHistory = this.data.transHistory
34. wx.showToast({
35. title: '删除成功',
36. icon: 'none',
37. duration: 1500
38. })
39. }
40. },
41. checkGroup(event) {
42. this.setData({
43. selectedHis: event.detail.value
44. })
45. console.log(this.data.selectedHis)
46. },
47. async delSelect() {
48. const selects = this.data.selectedHis
49. selects.sort()
50. console.log("删除记录" + selects)
51. const res = await wx.showModal({
52. title: '提示',
53. content: '是否删除历史记录',
54. })
55. if (res.confirm) {
56. for (var i = 0; i < selects.length; i++) {
57. this.delSingle(selects[i] - i)
58. }
59. app.globalData.transHistory = this.data.transHistory
60. wx.showToast({
61. title: '删除成功',
62. icon: 'none',
63. duration: 1500
64. })
65. }
66. },
67. multiSelect() {
68. this.setData({
69. multiSelectState: !this.data.multiSelectState
70. })
71. },
72. */\*\**
73. \* 生命周期函数--监听页面加载
74. \*/
75. onLoad(options) {
76. const res = wx.getStorageInfoSync();
77. for (let i = 0; i < res.keys.length; i++) {
78. let value = wx.getStorageSync(res.keys[i]);
79. const Arr = app.globalData.transHistory;
80. Arr.push({
81. id: i,
82. currentText: value.input,
83. outputText: value.output,
84. key: res.keys[i]
85. });
86. }
87. },
88. onShow() {
89. this.setData({
90. transHistory: app.globalData.transHistory
91. })
92. },
93. selectAll() {
94. let arr = this.data.transHistory
95. let selectArr=[]
96. let allSelected=this.data.allSelected
97. if(allSelected){
98. for (let i = 0; i < arr.length; i++) {
99. arr[i].checked = false
100. }
101. allSelected=false
102. }
103. else{
104. for (let i = 0; i < arr.length; i++) {
105. arr[i].checked = true
106. selectArr.push(i)
107. }
108. allSelected=true
109. }
110. this.setData({transHistory:arr,selectedHis:selectArr,allSelected:allSelected})
111. }
112. })

History.wxml：根据历史记录数组渲染每一条历史记录即可

1. <view class="topButton">
2. <view class="leftButton">
3. <block wx:if="{{multiSelectState}}">
4. <image class="singleTopButton" src="../../asset/pic/qx.png" mode="scaleToFill" bind:tap="selectAll" />
5. <image class="singleTopButton" src="../../asset/pic/sc.png" mode="scaleToFill" bind:tap="delSelect" />
6. </block>
7. </view>
8. <view class="rightButton">
9. <image class="singleTopButton" src="../../asset/pic/dx.png" mode="scaleToFill" bind:tap="multiSelect" />
10. </view>
11. </view>
12. <block wx:if="{{multiSelectState}}">
13. <checkbox-group bindchange="checkGroup">
14. <view class="singleLine" wx:for="{{transHistory}}" wx:key="id">
15. <checkbox class="checkBox" value="{{item.id}}" checked="{{item.checked}}" />
16. <singleHistory class="his" currentText="{{item.currentText}}" outputText="{{item.outputText}}" index="{{item.id}}"></singleHistory>
17. </view>
18. </checkbox-group>
19. </block>
20. <block wx:else>
21. <view wx:for="{{transHistory}}" wx:key="id">
22. <singleHistory currentText="{{item.currentText}}" outputText="{{item.outputText}} " index="{{item.id}}" bind:del="checkDel"></singleHistory>
23. </view>
24. </block>

History.scss：

1. .topButton{
2. height: 100rpx;
3. display: flex;
4. margin-right: 20rpx;
5. margin-left: 10rpx;
6. align-items:flex-end;
7. justify-content:space-between;
8. }
9. .leftButton{
10. display: flex;
11. align-items:flex-end;
12. justify-content: flex-start;
13. }
14. .rightButton{
15. display: flex;
16. align-items:flex-end;
17. justify-content: flex-end;
18. }
19. .singleTopButton{
20. height: 60rpx;
21. width: 60rpx;
22. margin-right:20rpx;
23. }
24. .singleLine{
25. margin-left: 10rpx;
26. display: flex;
27. align-items: center;
28. }
29. .checkBox{
30. flex: 1;
31. }
32. .his{
33. flex: 15;
34. }

#### pages/ClipPicture

图片裁剪界面主要实现的功能有：

* 接收翻译界面传入的图片URL
* 将裁剪后的图片URL传入app.globalData中

ClipPicture.js：点击裁剪按钮后，将调用reservePhoto将图片URL传入app.globalData

1. Page({
2. data: {
3. src: '',
4. width: 250, *//宽度*
5. height: 250, *//高度*
6. },
7. onLoad: function (options) {
8. *//获取到image-cropper实例*
9. this.cropper = this.selectComponent("#image-cropper");
10. *//开始裁剪*
11. this.setData({
12. src: options.photoPath
13. })
14. wx.showLoading({
15. title: '加载中'
16. })
17. },
18. cropperload(e) {
19. console.log("cropper初始化完成");
20. },
21. loadimage(e) {
22. console.log("图片加载完成", e.detail);
23. wx.hideLoading();
24. *//重置图片角度、缩放、位置*
25. this.cropper.imgReset();
26. },
27. clickcut(e) {
28. console.log(e.detail);
29. *//点击裁剪框阅览图片*
30. wx.previewImage({
31. current: e.detail.url, *// 当前显示图片的http链接*
32. urls: [e.detail.url] *// 需要预览的图片http链接列表*
33. })
34. },
35. reservePhoto() {
36. console.log("reservePhoto")
37. var app = getApp()
38. this.cropper.getImg((obj) => {
39. app.globalData.reservePhotoURL = obj.url;
40. wx.navigateBack({
41. delta: -1
42. })
43. });
44. }
45. })

ClipPicture.wxml：

1. <view class="textBody"><text class="text">点击裁剪框预览</text></view>
2. <view>
3. <image-cropper id="image-cropper" limit\_move="{{true}}" disable\_rotate="{{true}}" width="{{width}}" height="{{height}}" imgSrc="{{src}}" min\_width="{{1}}" min\_height="{{1}}"  bindload="cropperload" bindimageload="loadimage" bindtapcut="clickcut"></image-cropper>
4. </view>
5. <view class="buttonBody">
6. <button class="button" bind:tap="reservePhoto">裁剪</button>
7. </view>

ClipPicture.scss：

1. .mainBody{
2. display: flex;
3. background-color: aqua;
4. }
5. .buttonBody{
6. display: flex;
7. flex-direction: row;
8. position: absolute;
9. bottom: 100rpx;
10. z-index: 10;
11. height: 100rpx;
12. margin-left: 20rpx;
13. margin-right: 20rpx;
14. width:710rpx;
15. align-items: center;
16. justify-content:space-around;
17. }
18. .button{
19. height: 100rpx;
20. z-index: 10;
21. }
22. .textBody{
23. display: flex;
24. position: absolute;
25. top: 50rpx;
26. margin-left: 20rpx;
27. margin-right: 20rpx;
28. z-index: 10;
29. height: 100rpx;
30. width:710rpx;
31. justify-content: center;
32. align-items: center;
33. }
34. .text{
35. z-index: 10;
36. color:white;
37. }

## 系统界面展示

### TransLink翻译界面

在翻译界面中

* 用户可以在输入文本框中输入想要翻译的文本，文本下方会显示当前字数以及字数上限
* 用户也可以点击下方的播放按钮播放输入文本对应的语音
* 用户也可以点击复制按钮复制输入文本到剪切板
* 用户输入完毕后，可点击翻译按钮进行翻译，小程序会默认自动检测输入文本的语言，并翻译为中文，显示在下方的输出文本框
* 用户也可以点击语言来唤出语言选择框，选择语言后，小程序会按照用户选择的语言进行翻译
* 用户也可以点击切换按钮交换源语言和目标语言
* 用户可以点击输出文本框中的播放与复制按钮，效果与输入文本框中的一致，但是作用对象变成了输出文本
* 用户点击麦克风按钮后，小程序会开始录音，再次点击麦克风按钮后，小程序会结束录音，并将语言识别结果自动显示在输入文本框中
* 用户可以点击照相机按钮进入ClipPicture界面，在该界面确认图片后，小程序会将图片识别结果显示在输入文本框中

图6- 1翻译界面

### ClipPicture图片裁剪界面

在裁剪界面中，用户可以方便地截取想要翻译的内容

* 用户可以拖动裁剪框选择需要翻译的内容
* 图片裁剪界面会根据裁剪框在界面中的位置与大小，灵活地对裁剪区域进行缩放，以便用户能够清晰地看到裁剪区域的内容
* 确认无误后，用户可以点击裁剪按钮确认裁剪，随后小程序将跳转到翻译界面，图片识别的内容也会填充到输入文本框中

图6- 2图片裁剪界面

### History历史记录界面

在历史界面，用户可以查看本设备上的历史记录，并且可以进行删除历史记录的操作

* 用户可以单级某条历史记录，该记录会展开，显示完整的输入输出文本，并且带有播放和删除按钮
* 如果用户不需要该条记录，可以点击删除按钮，再次点击消息提示框中的确定按钮后，即可删除该条历史记录

图6- 3删除单条历史记录

* 用户也可以点击右上角的多选按钮激活左侧的复选框
* 用户可以自行选择需要删除的历史记录，也可以点击上方的全选按钮全选或者全不选历史记录
* 选择好历史记录后，可以点击删除按钮删除选择的历史记录

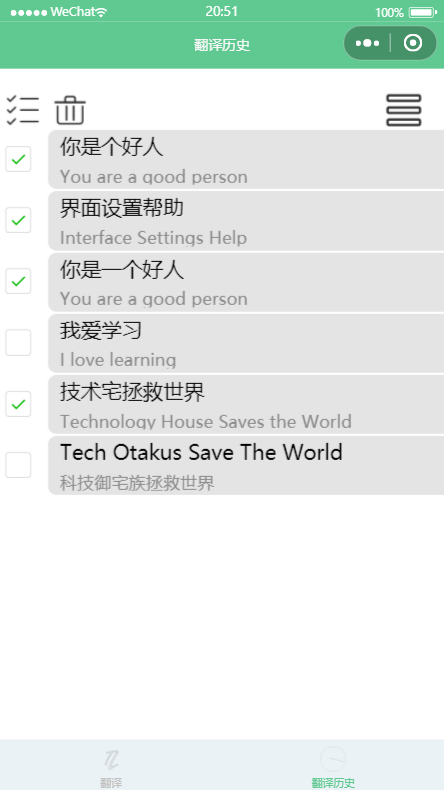
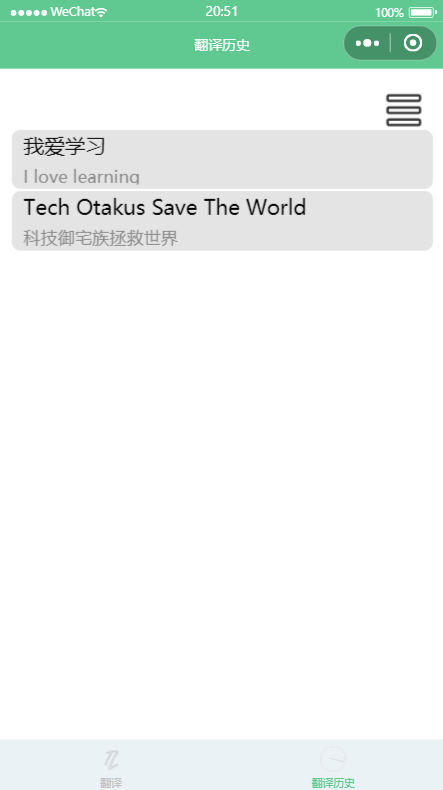
 

图6- 4删除多条历史记录

## 总结