虚拟形象文档

1 基于Flask的web服务器搭建

Flask是一个轻量级的基于Python的web框架。

1.1 框架创建&路由Route

Flask框架带有一个默认的static文件夹用以存放静态文件（比如css、js、图片、视频、音频等），和一个默认的templates文件夹用以存放html模板。

```
app=Flask(
            __name__,
            template_folder='xxx',
            static_folder='xxx',
            static_url_path='/xxx'
        )
```

1. template_folder ： 指定存放模板的文件夹的名称（默认为templates）
2. static_folder ： 指定存放静态文件资源的文件夹的名称（默认为static）
3. static_url_path ： 指定静态文件的访问路径（默认与static_folder一致）

```
from flask import Flask, render_template, request, jsonify

app = Flask(__name__, static_url_path="/static")
```

如上图所示，本项目采用默认配置创建了框架。
然后开始设置路由：

```
@app.route('/', methods=['GET', 'POST'])
# 定义应答函数，用于获取输入信息并返回相应的答案
def reply():
    if request.method == 'GET':
        return render_template('index.html')
    else:
    # 获取参数信息
        # 语句分词
        # 生成回答信息
        # unk值处理
        # 空处理
```

```
return render_template('index.html', response=res_msg)
```

该语句表示如果是GET请求访问网页的话,就在浏览器上显示’index.html’页面。如果用户点击了录制提问的按钮则POST经过语音录制、分词、匹配、unk值处理等一系列步骤生成回答信息并由’index.html’（内的文本框）负责接收并显示。这里的返回’render_template’函数即是在调用templates文件夹内的html模板。

1.2 模板编写&虚拟形象

以下是网页界面模板：

```
<head>
      <meta charset="UTF-8">
      <title>BIT智能问答系统</title>
      <link href="../static/common.css" rel="stylesheet">
      <style>
            html,
            body {
                  background: #F7EED6;
                  height: 100%;
                  width: 100%;
                  margin: 0px;
                  padding: 0px;
                  text-align: left;
            }
      </style>
</head>
<body>
<div id="top">
      <h1>BIT智能问答系统</h1>
<div>
<div id="left">
      <h2 class="s1" style="color:#FFA700; margin-left:10px">你想问点什么？</h2>
      <form action="/" method="post">
            <input class="btn_a" type="submit" value="点击此键开始录音">
      </form>
</div>
<div id="right">
      <h2 style="color:#FFA700; margin-left:75px">我的回答：</h2>
      <form action="/" method="post" class="s1" align="left">
            <textarea rows="20" cols="50" font-size="" name="txt">{{ response }}
</textarea>
      </form>
</div>
<img src="\static\images\BITpng.png" width="150px" height="150px" align="left">
</body>
```

主要是网页的展示结构和两个控件（提问录音按钮&回答展示框）。

然后是虚拟形象的实现部分，本项目采用了最基础的L2D的默认model和动作。对于用户的操作能做出一定的回应。但局限于L2D有限的web端接口和扩展能力，目前为止并不能很好地实现原本预想的自设多个动作接口用以丰富用户交互的初衷。这部分是本项目的一个遗憾：

```
<script type="text/javascript" charset="utf-8"   src="\static\live2d-widget.js-master\lib\L2Dwidget.0.min.js"></script>
<script type="text/javascript" charset="utf-8"   src="\static\live2d-widget.js-master\lib\L2Dwidget.min.js"></script>
<script type="text/javascript">
      L2Dwidget.init(
           {
            "display": {
                  "superSample": 2,
                  "width": 200,
                  "height": 400,
                  "position": "right",
                  "hOffset": 520,
                  "vOffset": 0
                },
            "dialog": {
                  "enable": true,
                  "script": {
                        'every idle 10s': '您好，BIT智能问答系统为您服务！',
                        'tap body': '您、您好……',
                        'tap face': '欢、欢迎……'
                       }
                }
           }
      );
</script>
</html>
```

L2Dwidget.0.min.js和 L2Dwidget.min.js是支持L2D的脚本。

（脚本源码在https://github.com/xiazeyu/live2d-widget.js.git，且在git中不能直接找到该脚本文件——L2Dwidget.min.js 脚本是需要编译生成的。

在\static\live2d-widget.js-master文件目录下通过命令cnpm install安装依赖包，然后cnpm run build:dev执行编译项目的脚本，运行完毕后lib目录生成，L2Dwidget.min.js和L2Dwidget.0.min.js脚本也成功生成。）

【L2Dwidget.init();】是初始化L2D模型的语句。默认加载的是live2d-widget-model-shizuku这个模型，在live2d-widget.js-master/src/config/defaultConfig.js中可以看到默认配置如下：

```
const defaultConfig = {
   model: {
```

```
        jsonPath: 'https://unpkg.com/live2d-widget-model-shizuku@latest/assets/shizuku.model.json',
        scale: 1,
    },
    display: {
        superSample: 2,
        width: 200,
        height: 400,
        position: 'right',
        hOffset: 0,
        vOffset: -20,
    },
    mobile: {
        show: true,
        scale: 0.8,
        motion: true,
    },
    name: {
        canvas: 'live2dcanvas',
        div: 'live2d-widget',
    },
    react: {
        opacity: 1,
    },
    dev: {
        border: false
    },
    dialog: {
        enable: false,
        script: null
    }
}
```

在 src/index.js 文件中可以看到配置的说明如下：

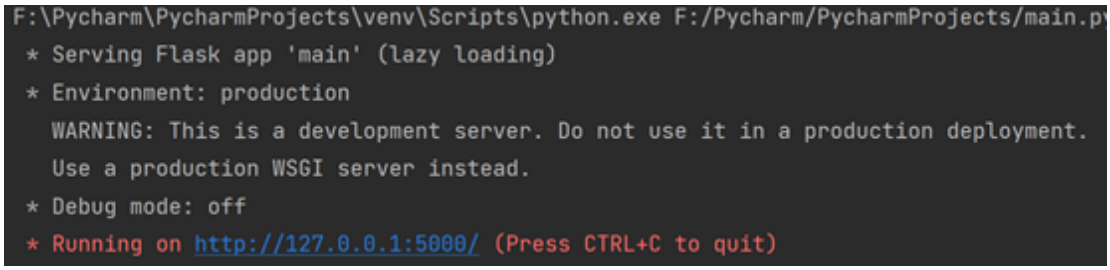
```
/**
 * The init function
 * @param {Object}    [userConfig] User's custom config 用户自定义设置
 * @param {String}    [userConfig.model.jsonPath = ''] Path to Live2D model's main json eg. `https://test.com/miku.model.json` model主文件路径
 * @param {Number}    [userConfig.model.scale = 1] Scale between the model and the canvas 模型与canvas的缩放
 * @param {Number}    [userConfig.display.superSample = 2] rate for super sampling rate 超采样等级
 * @param {Number}    [userConfig.display.width = 150] Width to the canvas which shows the model canvas的长度
 * @param {Number}    [userConfig.display.height = 300] Height to the canvas which shows the model canvas的高度
```

```
 * @param {String}    [userConfig.display.position = 'right'] Left of right side to show 显示位置：左或右
 * @param {Number}    [userConfig.display.hOffset = 0] Horizontal offset of the canvas canvas水平偏移
 * @param {Number}    [userConfig.display.vOffset = -20] Vertical offset of the canvas canvas垂直偏移
 * @param {Boolean}   [userConfig.mobile.show = true] Whether to show on mobile device 是否在移动设备上显示
 * @param {Number}    [userConfig.mobile.scale = 0.5] Scale on mobile device 移动设备上的缩放
 * @param {String}    [userConfig.name.canvas = 'live2dcanvas'] ID name of the canvas canvas元素的ID
 * @param {String}    [userConfig.name.div = 'live2d-widget'] ID name of the div div元素的ID
 * @param {Number}    [userConfig.react.opacity = 0.7] opacity 透明度
 * @param {Boolean}   [userConfig.dev.border = false] Whether to show border around the canvas 在canvas周围显示边界
 * @param {Boolean}   [userConfig.dialog.enable = false] Display dialog 显示人物对话框
 * @param {Boolean}   [userConfig.dialog.hitokoto = false] Enable hitokoto 使用一言API
 * @return {null}
 */
```

于是参照配置说明，设置dialog为true并以此作为交互方式之一。
以上是网页模板的全部内容。

1.3 测试运行：

测试运行图片如下：



BIT智能问答系统

你想问点什么?

点 击 此 键 开 始 录 音

我的回答: