Text To Speech Proof Of Concept

Developers Guide

# UpStage Project Semester 2 2014 Author: James Williams Date: 28/10/2014

# Proof of concept for Text To Speech using HTLM5, Python, Flask over a LAMP stack (Linux, Apache, MySQL, PHP).

One of our main objectives this semester was to identify which technologies are used and how they are implemented in UpStage currently. I have determined that the current Text To Speech Engine appears to be called Festival (using Festival, EMBROLA and eSpeak libraries for voices) which generates .mp3 audio files in the directory path “UpStage2013\server\src\html\speech.py” named “utter-” followed by an incrementing number. The JavaScript file “UpStage2013\server\src\html\script\mediaupload.js” calls “FlowPlayer” to play the audio files in the client’s browser which uses Flash. Flash makes the application unfavourable in terms of today’s technology limitations and uses. What I’m talking about is the fact that Flash is not supported for browsers on mobile devices. If I try to test an avatar’s voice from Chrome on my android phone a message appears telling me I need to install Flash to be able to do that.

Another reason the Text to Speech engine/delivery system should be replaced is the voices are of low quality and seem to work and stop working randomly causing reliability issues for the application.

The concept I developed and experimented with was to generate text to speech audio files on the server much the same way the current software does, but then to deliver the file straight to the HTML using the HTML5 <audio> tag to be played in the client, i.e. the user’s browser.

I succeeded somewhat in getting this to work. Although the application is an early prototype, it demonstrates that HTML5 is suitable for delivering audio files from a server to browsers not just on PCs but on mobile devices.

My proof of concept utilises HTLM5, Python, and Flask running on within an Apache2 server on Linux. The Linux distribution I used was Ubuntu but any distribution should be fine as long as it can install the necessary programs.

To run this proof of concept I suggest installing the latest stable release of Ubuntu (or other distro) either on a physical drive or perhaps more conveniently inside a virtual machine (for example using the free program VirtualBox).

First of all if you aren’t familiar with or haven’t set up a LAMP stack, set one up following this guide: <https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu>

Secondly, install and set up the virtual development environment within Linux for which you’ll use to develop the FlaskApp (This proof of concept) using the following guide: <https://www.digitalocean.com/community/tutorials/how-to-deploy-a-flask-application-on-an-ubuntu-vps>

Once that is done all you need to do is insert the files from my proof of concept folder, copying the directory structure they’re in. The python file \_\_init\_\_.py should be in the second FlaskApp folder “/var/www/FlaskApp/FlaskApp” and the test.html file should be in the static folder, “/var/www/FlaskApp/FlaskApp/static.” It should be easy enough to follow if you finished the above tutorials. Then, all you need to do to launch the app is to run “sudo python \_\_init\_\_.py“ in your terminal with the virtual environment activated.

Now if you browse to “localhost:5000” on the linux machine or browse to the linux machine’s IP address followed by “:5000” you should be greeted by a text input, a “Generate File” button and a “Play File” button. You should now be able to type something in the box, press the generate file button, and then pressing the play file button should play back a voice saying the words you typed in. This should work on any browser that supports wav files (that is any browser except IE). Please see the following link to check which audio file types are support in which browser: <http://www.w3schools.com/tags/tag_audio.asp>

## Advice for future UpStage Developers:

The prototype is still in the very early stages and there is a problem to do with caching. If you type something else into the text box and generate another audio file, the original file will be overwritten as the filename is hardcoded in the python script which you can verify by playing the file on your Linux machine however the browser will still have the original file cached. Even if the source is updated each time the button is pressed (via HTML/JavaScript) it will play the file that it has cached. This can be overcome in two different ways. The first way which I do not recommend for user friendliness reasons is to remove caching from the Apache server so each client has to load everything completely from scratch all the time. The more practical solution is to append something to the filename each time a new text to speech file is created. This is how the current UpStage software engine works and ideally how it should continue to work. I was unable to get the incremental naming scheme working in time. While I was trying to implement it I began to have doubts that Flask is the right web framework. Flask seemed to need to redirect the webpage each time a call was made in order to run Python code which made it very difficult for me to find a way of holding a count in JavaScript (to append to the filename). Maybe django would be more suitable but further research would need to be carried out to determine the most suitable framework. Unfortunately I did not have enough time to research other Python web frameworks before needing to release the proof of concept along with this companion document.

## The Flask disclaimer:

N.B. From the Flask website <http://flask.pocoo.org/docs/0.10/license/> :

”Flask is licensed under a three clause BSD License. It basically means: do whatever you want with it as long as the copyright in Flask sticks around, the conditions are not modified and the disclaimer is present. Furthermore you must not use the names of the authors to promote derivatives of the software without written consent.”

And the bottom of that page reads:

“Note: we would appreciate that you make the image a link to [http://flask.pocoo.org](http://flask.pocoo.org/)/ if you use it on a web page.”