Text Sync Tool - PHP

See file: **txt_sync.php**

This script requires a web server running PHP version 5.1.6 or newer, using the SimpleXML extension.

Expandable

This script has a minimal structure that is versatile and intended to be modified, extended and adapted to the framework of any project.

For example, three variables are assigned values at the start of the script:

```
$media_file = "example_audio.mp3";

$eaf_file = "example_elan.eaf";

$player_title = "When Timothy Fell in The Latrine";
```

These values could be delivered by POST or GET as part of a dynamic process by which this single script would transcribe and present HTML from various ELAN files, chosen from a list through a form interface.

XML Parsing into HTML

Once the ELAN (XML) is loaded, the script loops through all of the <TIME_ORDER> <TIME_SLOT> tags and pushes the *TIME_VALUE* attribute values into a one dimensional array:

```
foreach ($xml->TIME_ORDER->TIME_SLOT as $time_slot)
{
        $time_slot_array[] = $time_slot['TIME_VALUE'];
}
```

Next it loops through the <TIER> tags looking for those with a *LINGUISTIC_TYPE_REF* attribute value of “transcription” or “gloss”.

The value of the <TIER> *PARTICIPANT* attribute (the name of the person speaking) is split at blank spaces and the first initials are extracted for CSS color-coded display in the speaker key and transcript display area (see HTML and CSS documentation).

```
$speaker = $a_tier['PARTICIPANT'];

$speaker_parts = explode(" ", $speaker);

$spkr = "";

foreach($speaker_parts as $sp_prt) { $spkr .= substr($sp_prt, 0, 1); }

$tier_css = " tr" . $tier_count++;

$tier_list .= "<li><span class=\"spkr_key " . trim($tier_css) . "\">" . $spkr .
"</span><span> &middot; </span><span class=\"spkr_name\">" . $speaker . "</span></li>\n";
```

If the <TIER> *LINGUISTIC_TYPE_REF* attribute value is “transcription” the script loops through the <ANNOTATION> tags, extracting the *ANNOTATION_ID* attribute value and the integer portion of the *TIME_SLOT_REF1* and *TIME_SLOT_REF2* attribute values of the current <ALIGNABLE_ANNOTATION> tag.

The *TIME_SLOT_REF1* and *TIME_SLOT_REF2* values become the keys to look up the *TIME_VALUE* attribute values stored previously in the $time_slot_array[].

This array is indexed from 0, so 1 is subtracted from the key.

The values in the array are in milliseconds, so they are divided by 1000.

The resulting values, in seconds, become the *data-start* and *data-stop* attribute values of the individual <li> tags in the transcript display area of the HTML.

The transcript text is extracted from the <ANNOTATION_VALUE> tag.

A string of HTML is assembled using the values found in each loop.

This string is pushed into an associative array using the *TIME_SLOT_REF1* attribute value as the key for subsequent sorting before being echoed as HTML.

```
foreach ($a_tier->ANNOTATION as $a_nnotation)
{
	$time_start_ref = (int) substr($a_nnotation->ALIGNABLE_ANNOTATION['TIME_SLOT_REF1'], 2);
	$time_stop_ref = (int) substr($a_nnotation->ALIGNABLE_ANNOTATION['TIME_SLOT_REF2'], 2);
	$line_id = $a_nnotation->ALIGNABLE_ANNOTATION['ANNOTATION_ID'];

	$resulting_span_string = "<li class=\"txt_ln" . $tier_css . "\" data-start=\"" .
	$time_slot_array[$time_start_ref-1]/1000 . "\" data-stop=\"" .
	$time_slot_array[$time_stop_ref-1]/1000 . "\"><span class=\"spkr\">" . $spkr .
	"</span><span> : </span><span class=\"spkn\" id=\"" . $line_id . "\">" .
	htmlspecialchars($a_nnotation->ALIGNABLE_ANNOTATION->ANNOTATION_VALUE) .
	"</span></li>\n";

	addArray($output_array, $time_start_ref, $resulting_span_string);
}
```

Note: the addArray() function, used to build the associative array, is defined at the top of the script:

```
	function addArray(&$array, $id, $var)
	{
		$tempArray = array($var => $id);
		$array = array_merge($array, $tempArray);
	}
```

If the <TIER> *LINGUISTIC_TYPE_REF* attribute value is “gloss” the process is similar; though the output strings do not require sorting because they become HTML tags that are hidden/shown individually by the JavaScript (see HTML and JavaScript documentation).

The script loops through the <ANNOTATION> tags, extracting the *ANNOTATION_REF* attribute value from the <REF_ANNOTATION> tag and the transcript text from the <ANNOTATION_VALUE> tag.

A growing string of HTML is concatenated from the values found in each loop.

```
foreach ($a_tier->ANNOTATION as $a_nnotation)
{
	$line_ref = $a_nnotation->REF_ANNOTATION['ANNOTATION_REF'];
	$line_value = $a_nnotation->REF_ANNOTATION->ANNOTATION_VALUE;
	$line_out = htmlspecialchars($line_value);
	$spkr_out = $spkr;
	$gloss_tier_string .= "<div class=\"txt_ref\" id=\"r" . $line_ref . "\"><span
	class=\"spkr\">" . $spkr_out . "</span><span> : </span><span class=\"tran\">" . $line_out
	. "</span></div>\n";
}
```

Finally the HTML for the media player is generated and all the assembled and sorted strings are echoed out to the HTML.

Further Possibilities

Note the inclusion of these three variables:

```
	$start_at_time = 0;

	$start_at_time_end = 0;

	$specific_start_line_id = "x0";
```

Within the loop through the <ANNOTATION> tags of a “transcription” <TIER>, the value assigned to **$specific_start_line_id** is compared to the value of the *ANNOTATION_ID* attribute of the current <ALIGNABLE_ANNOTATION> tag:

```
	$line_id = $a_nnotation->ALIGNABLE_ANNOTATION['ANNOTATION_ID'];

	if ($line_id == $specific_start_line_id)
	{
		$start_at_time = $time_slot_array[$time_start_ref-1]/1000;
		$start_at_time_end = $time_slot_array[$time_stop_ref-1]/1000;
	}
```

The values assigned to **$start_at_time** and **$start_at_time_end** are echoed to the HTML as values for two variables used by the JavaScript to scroll to and play through to the end of a specific <li> of transcribed text (see **txt_sync.js** and JavaScript documentation)

```
	<script type="text/javascript">
		var initial_time = <?php echo $start_at_time; ?>;
		var initial_time_end = <?php echo $start_at_time_end; ?>;
	</script>
```

This potential is not used in the script process as it is in the example files; however with some modification it would be possible to pass a value to **$specific_start_line_id** by POST or GET.