UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

# WEB SPEECH RECOGNITION & TRANSLATION

## Web application per il riconoscimento vocale e la traduzione

API web speech recognition Google – API Microsoft Translator


Sistemi per la collaborazione in Rete

A.A 2013-2014

**Prof.**

F.Lanubile

F.Calefato

**Studenti:**

F.Giannini matr. 609499

G.Loviglio matr.614547

# Indice

# 1. Introduzione

Nell'era dei social e dei potentissimi dispositivi mobile, smartphone e tablet, cresce in maniera esponenziale la necessità di comunicare tra gli individui per ampliare il proprio bagaglio sociale e culturale.

Gli ostacoli maggiormente presenti tuttavia, sono rappresentati sia dalla limitata conoscenza di lingue che dalla frequenza, come spesso accade, di errori lessicali e grammaticali nella scrittura di testi nella propria lingua madre.

Inoltre, la delocalizzazione porta a connettere comunità di persone dislocate in varie aree geografiche del mondo, che hanno la necessità di interfacciarsi per il coordinamento dei lavori, con conseguenti problematiche legate alle differenze linguistiche e comunicative.

Per risolvere questi problemi pertanto, è indispensabile l'utilizzo dapprima di un riconoscitore vocale ed in seguito di un traduttore.

Abbiamo sviluppato quindi, un'applicazione desktop che grazie alle API messe a disposizione da Google(riconoscitore vocale) e Microsoft (traduttore), è in grado di superare queste problematiche in maniera semplice, intuitiva ed interessante. Inizialmente, l'applicazione doveva essere implementata e sviluppata completamente con le API di Google. Poiché le API per la traduzione richiedono una chiave a pagamento, abbiamo virato la nostra scelta sull'utilizzo dell' API gratuita messa a disposizione da Microsoft.

L'applicazione è creata per poter essere eseguita sul **browser Chrome di Google**.

# 2. Web Speech Recognition API Google

Web speech recognition API (Application Program Interface) permettono, avendo a disposizione un web browser, di acquisire un input vocale e fornire come output il testo corrispondente alla sintesi vocale riconosciuta (utilizzando riconoscitori standard o software screen-reader, questo lavoro è impossibile). L' API stessa è un metodo agnostico di riconoscimento del parlato che può supportare sia sistemi server-based che client-based/embedded.

Essa è progettata per identificare sillabe, singole frasi come anche discorsi complessi ma continuativi.

Il riconoscimento vocale, la cui precisione e velocità nella individuazione della corretta interpretazione è strettamente proporzionale alla speditezza e alla potenza della connessione, viene fornito alla pagina come una lista di ipotesi, alle quali vengono matchate altre informazioni pertinenti. Questi parametri aiutano le API a fornire un riconoscimento più accurato. La stringa, in questo frangente, è visualizzata nel colore grigio chiaro e viene modificata in base a quanto appreso dal sistema e alle parole pronunciate in seguito, a meno di pause eccessivamente lunghe (influenza contestuale), fin quando il sistema non convalida l'ipotesi più accreditata. A questo proposito verrà utilizzata la variabile "interim_transcript" che per ovvie ragioni di riconoscimento, non ha lunghezza fissa.

Entro pochi secondi, la stringa diventa definitiva e viene visualizzata in colore nero. Da quel momento, qualunque altra parola o frase pronunciata successivamente a quella che è stata resa "definitiva", non inficerà in nessun modo sulla precedente. La stringa definitiva viene memorizzata nella variabile

“final_transcript”. Essa, come “interim_transcript”, non ha lunghezza fissa, ma viene incrementata fino a che ci saranno elementi da riconoscere o non venga cliccato il pulsante che esprime la volontà di terminare il riconoscimento vocale.

Come ogni sistema web-based, anch’ esso è soggetto ad attacchi che minano la sicurezza interna. È possibile infatti che hacker intercettino il parlato e oscurino l’output (o lo falsifichino), per avere accesso ad informazioni private. Per questo è altamente sconsigliato utilizzare un’applicazione di traduzione per password o dati sensibili.

# 3. Microsoft Translator API

Microsoft Translator è un servizio host accessibile via API che permette la traduzione di applicazioni senza l'ausilio di un traduttore umano. È utilizzato in qualsiasi contesto ove si necessiti di una traduzione, dalla realizzazione di applicazioni per smartphone e tablet, ad applicazioni desktop, pagine web ecc… . La traduzione di siti può essere effettuata in due modi:

- Utilizzando il widget Microsoft Translator
- Attraverso interfacce SOAP, http e Ajax

Nel nostro progetto si è fatto uso delle API di Microsoft attraverso il protocollo http dove, oltre alle funzioni implementate per effettuare la traduzione, si sono attuate le fasi preliminari per ottenere le credenziali token, necessarie ed indispensabili per poter utilizzare le API.

# 4. Creazione dell’account sul Marketplace di Windows Azure.

Condizione necessaria e sufficiente per poter ottenere l’accesso alle API di Microsoft è quella di dotarsi di un account Microsoft, con il quale sarà possibile accedere oltre ai servizi Outlook.com, Skype ecc… al Marketplace Microsoft Azure. Quindi dopo essersi iscritti all’indirizzo http://login.live.com, si effettua il login all’indirizzo https://datamarket.azure.com/. A questo punto si dovranno seguire i seguenti passi:

- Iscriversi all’API utilizzando l’account con cui si è registrati.
- Iscrivere la propria applicazione sul Marketplace.
- Ottenere le credenziali Client ID e Client Secret dell’applicazione che è stata registrata.

Le credenziali devono essere ricordate in quanto saranno inserite nel file credential.php che permette quindi l’accesso all’API di traduzione.

# 5. Implementazione dell'applicazione.

In questa sezione verranno presentati i moduli che compongono l'applicazione. La cooperazione di questi, darà origine alla nostra web application.

## 5.1 Modulo 'Credential.php'

Questo modulo contiene le credenziali di accesso per l'utilizzo delle API di Microsoft. Le credenziali ottenute in fase di registrazione sono:

Client ID, Client Secret, OAuth URL, Application scope URL e l'application grant.

Ciascun valore è stato salvato all'interno di una variabile PHP; si avvalorano quindi le variabili con i valori acquisiti in sede di registrazione dell'applicazione:

```php
<?php

    //Client ID of the application.
    $clientID = "SpeechRecognitionLoviglioGiannini";

    //Client Secret key of the application.
    $clientSecret = "XVofSDkfa6f/vUvebKznXSvNRz7+O2xRm3N4RVPQRwg=";

    //OAuth Url.
    $authUrl = "https://datamarket.accesscontrol.windows.net/v2/OAuth2-13/";

    //Application Scope Url
    $scopeUrl = "http://api.microsofttranslator.com";

    //Application grant type
    $grantType = "client_credentials";

?>
```

**Credential.php**

**(fig.1)**

Come è possibile notare, sono stati associati a ciascuna variabile i relativi valori. Si è deciso di creare un modulo a se stante sia per salvaguardare la modularità del progetto che l'eventuale modifica futura di questi valori. Inoltre, le variabili *'$clientID'* e *'$clientSecret*" rivestono notevole importanza in quanto sono univoci dell'autore dell'applicativo e pertanto devono rimanere segreti all'utilizzatore dell'applicazione.

## 5.2 Modulo 'AccessTokenAuthenication.php'

L'API di Microsoft Translator, per essere utilizzata, richiede l'autenticazione al Windows Azure Marketplace. Con questo modulo si crea la classe AccessTokenAutenthication in cui si va ad implementare la funzione di generazione del token di accesso, che sarà mandato al servizio Microsoft Translator così da autorizzare la traduzione del testo o della parola.

```php
<?php
    class AccessTokenAuthentication
{
   function getTokens($grantType, $scopeUrl, $clientID, $clientSecret, $authUrl)
    {
        try
        {
            $ch = curl_init();
            $paramArr = array ('grant_type'=>$grantType, 'scope'=>$scopeUrl, 'client_id'=>$clientID, 'client_secret'=>$clientSecret);
            $paramArr = http_build_query($paramArr);
            curl_setopt($ch, CURLOPT_URL, $authUrl);
            curl_setopt($ch, CURLOPT_POST, TRUE);
            curl_setopt($ch, CURLOPT_POSTFIELDS, $paramArr);
            curl_setopt ($ch, CURLOPT_RETURNTRANSFER, TRUE);
            curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);
            $strResponse = curl_exec($ch);
            $curlErrno = curl_errno($ch);
            if($curlErrno)
            {
                $curlError = curl_error($ch);
                throw new Exception($curlError);
            }
            curl_close($ch);
            $objResponse = json_decode($strResponse);
            if (isset($objResponse -> error))
            {
                throw new Exception($objResponse -> error_description);
            }
            return $objResponse -> access_token;
        } catch (Exception $e)
            {
                echo "Exception-".$e -> getMessage();
            }
    }
}

?>
```

**AccessTokenAutentication.php**

**(fig.2)**

Dopo aver definito il nome della classe, si dichiara la funzione getTokens che prende in input le credenziali del modulo ‘Crendential.php’. Nella funzione viene, innanzitutto, inizializzata una sessione cURL utilizzando il comando curl_init(). Esso appartiene alla libreria libcurl, libreria free lato client, che permette il trasferimento dell’URL. Supporta servizi come http, https, file, gopher, pop3, ecc. Inoltre supporta i certificati SSL, http POST, http PUT, proxy, autenticazione con nome utente e password (Digest, Kerberos, ecc.).

Successivamente si inizializza l’array contenente le credenziali di accesso passate come input alla funzione. Il contenuto dell’array verrà utilizzato dalla funzione http_build_query() che consente di generare una query http. Dopo aver impostato le opzioni per il trasferimento del cURL, utilizzando la funzione curl_setopt, il cURL viene eseguito. Dopo aver controllato che l’esecuzione sia avvenuta senza errore, si chiude la sessione cURL (chiamando la funzione curl_close()) ed infine si decodificala stringa JSON ricevuta. Si effettua un ulteriore controllo, questa volta sulla stringa JSON e in caso di errore viene sollevata un’eccezione, altrimenti viene restituito il token di accesso.

## 5.3 Modulo ‘HttpTranslator.php’

In questo modulo la classe helper “HTTPTransaltor” ha il compito di gestire la comunicazione con le API. Viene utilizzata la funzionalità cURL di PHP così la comunicazione avviene in maniera sincrona tramite il servizio web http.

```php
<?php


Class HTTPTranslator
{

    function curlRequest($url, $authHeader)
    {
        //Initialize the Curl Session.
        $ch = curl_init();
        //Set the Curl url.
        curl_setopt ($ch, CURLOPT_URL, $url);
        //Set the HTTP HEADER Fields.
        curl_setopt ($ch, CURLOPT_HTTPHEADER, array($authHeader,"Content-Type: text/xml"));
        //CURLOPT_RETURNTRANSFER- TRUE to return the transfer as a string of the return value of curl_exec().
        curl_setopt ($ch, CURLOPT_RETURNTRANSFER, TRUE);
        //CURLOPT_SSL_VERIFYPEER- Set FALSE to stop cURL from verifying the peer's certificate.
        curl_setopt ($ch, CURLOPT_SSL_VERIFYPEER, False);
        //Execute the  cURL session.
        $curlResponse = curl_exec($ch);

        //Get the Error Code returned by Curl.
        $curlErrno = curl_errno($ch);
        if ($curlErrno)
        {
            $curlError = curl_error($ch);
            throw new Exception($curlError);
        }

        //Close a cURL session.
        curl_close($ch);
        return $curlResponse;
    }
}

?>
```

**HttpTranslator.php**

**(fig.3)**

La funzione prende in input il request URL, il request header e i dati mandati attraverso il metodo POST dal form e restituisce il cURL di risposta o di errore.

A questo punto possiamo implementare le funzioni necessarie per la traduzione.

## 5.4 Modulo “index.php”

Il seguente è il modulo “principe” dell’applicazione. In esso sono contenuti sia le componenti grafiche che l’implementazione del funzionamento del sistema. Possiamo notare quindi, due sezioni principali:

1) Php: indicato tra i tag “<?php  ?>”, presenta la parte server dell’applicazione. In essa viene effettuata la fase di traduzione e la generazione dello stream audio per il text-to-speech.
2) HTML- JAVASCRIPT: racchiusa tra i tag “<html> </html>” e “<script> </script>”, è la parte client dell’applicativo. Contiene la definizione dei bottoni, textarea, oltre all’implementazione della sezione riguardante il riconoscimento vocale.

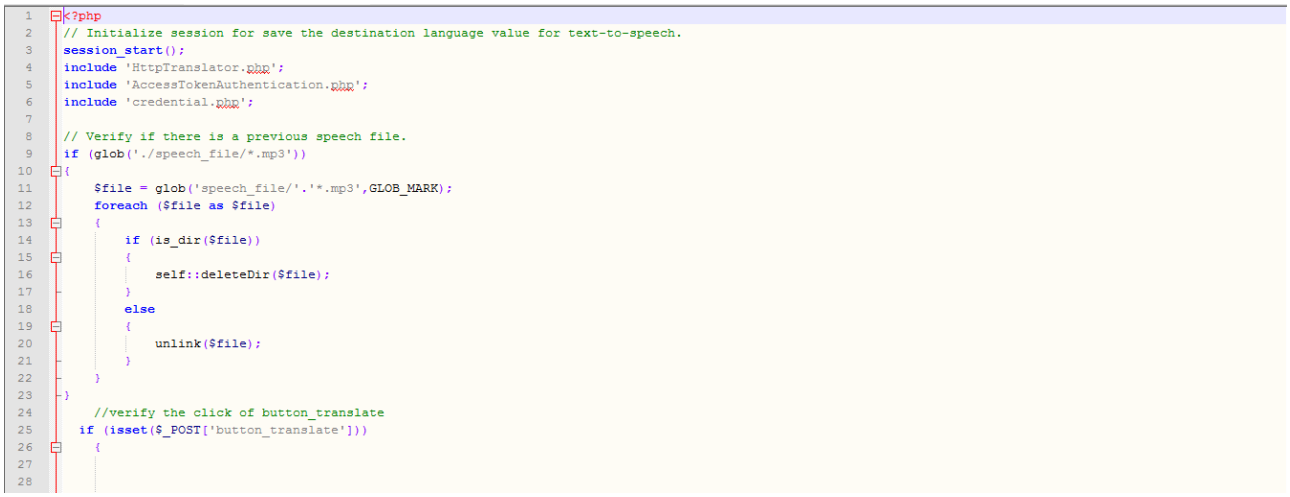
Analizziamo nel dettaglio il codice sorgente. In seguito verrà mostrato come i moduli cooperano tra di loro in fase di esecuzione.

### 5.4.1 PHP:

Innanzitutto si effettua il passaggio del testo e delle lingue sorgenti e destinazione (con relativi dialetti) per poter utilizzare le API di Microsoft e per effettuare, quindi, la traduzione e il text-to-speech. Si effettua, dunque, un controllo sulla presenza di precedenti file audio. Nel caso in cui sia presente già un file, esso viene eliminato, altrimenti si prosegue andando ad includere i moduli creati precedentemente. Successivamente vengono salvati in delle variabili di sessione le due lingue selezionate e i relativi dialetti; questo per permettere di salvare il contenuto delle variabili anche dopo il refresh della pagina. Le variabili di sessione salvano i dati sul server stesso ma si servono comunque di un cookie(la cui spiegazione verrà espletata successivamente), il SID.

Dalla teoria, la creazione di una sessione avviene con il comando session_start() che deve essere posizionato al primo rigo per evitare comportamenti anomali dello script.

L’inizializzazione della sessione, avviene infatti, alla riga 3 della nostra applicazione.



```php
<?php
// Initialize session for save the destination language value for text-to-speech.
session_start();
include 'HttpTranslator.php';
include 'AccessTokenAuthentication.php';
include 'credential.php';

// Verify if there is a previous speech file.
if (glob('./speech_file/*.mp3'))
{
    $file = glob('speech_file/'.'*.mp3',GLOB_MARK);
    foreach ($file as $file)
    {
        if (is_dir($file))
        {
            self::deleteDir($file);
        }
        else
        {
            unlink($file);
        }
    }
}
    //verify the click of button_translate
  if (isset($_POST['button_translate']))
    {
```

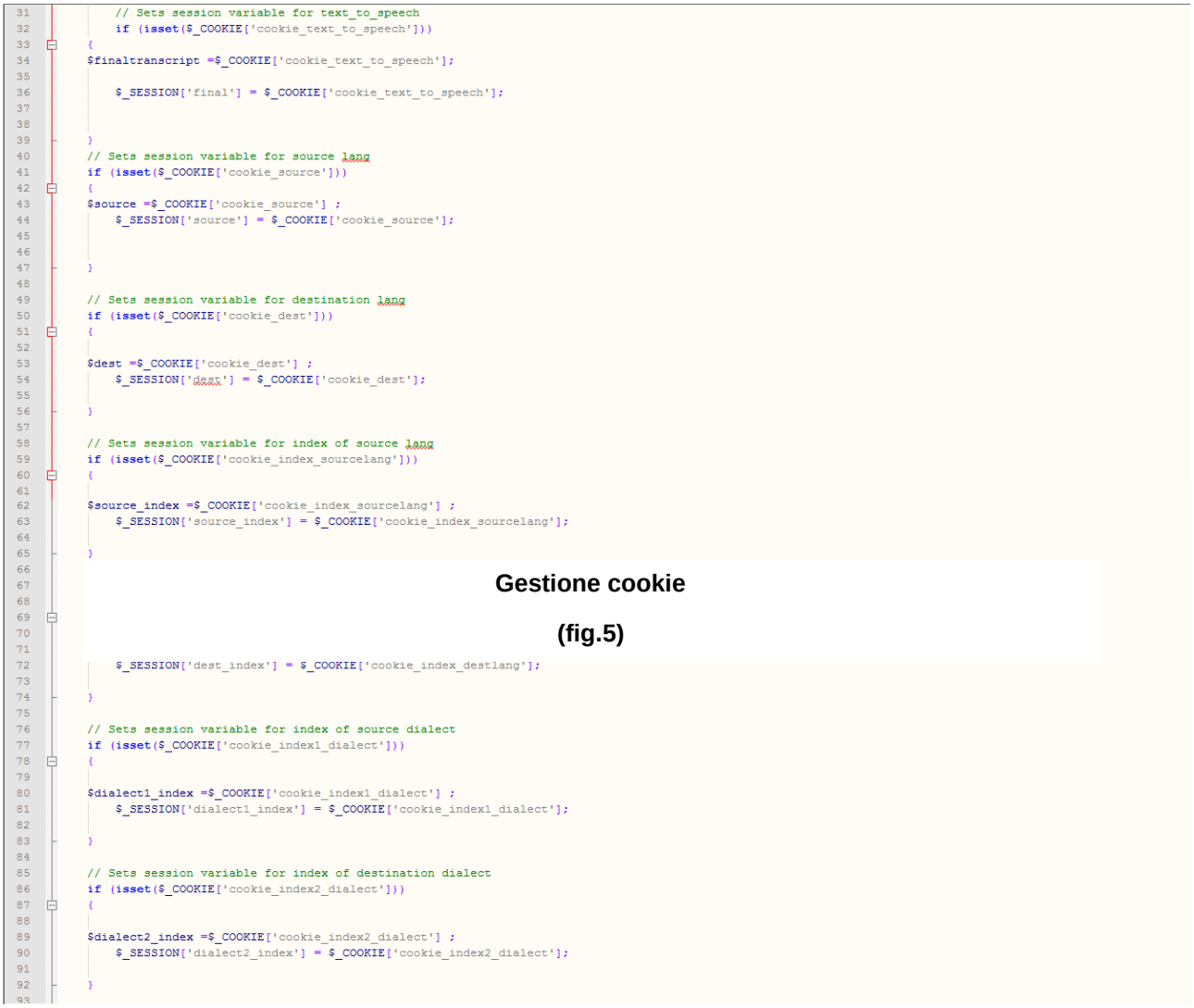
**Avvio sessione**

**(fig.4)**

Una volta avviata la sessione potranno essere create variabili di sessione utilizzando la variabile $_SESSION, in particolare si tratta di una array a cui possibile attribuire le chiavi che desideriamo. Nell’applicazione, le variabili di sessione devono corrispondere ai valori dei cookie che verranno creati nella sezione javascript (la creazione dei cookie verrà mostrata in seguito).

Lo standard di definizione utilizzato è il seguente:

```
    if (isset($_COOKIE['nome_cookie’]))
{
    $variabile_php =$_COOKIE['nome_cookie'];

$_SESSION['nome_sessione']= $_COOKIE['nome_cookie'];

}
```



```php
    // Sets session variable for text_to_speech
    if (isset($_COOKIE['cookie_text_to_speech']))
{
$finaltranscript =$_COOKIE['cookie_text_to_speech'];

    $_SESSION['final'] = $_COOKIE['cookie_text_to_speech'];


}
// Sets session variable for source lang
if (isset($_COOKIE['cookie_source']))
{
$source =$_COOKIE['cookie_source'] ;
    $_SESSION['source'] = $_COOKIE['cookie_source'];


}

// Sets session variable for destination lang
if (isset($_COOKIE['cookie_dest']))
{

$dest =$_COOKIE['cookie_dest'] ;
    $_SESSION['dest'] = $_COOKIE['cookie_dest'];

}

// Sets session variable for index of source lang
if (isset($_COOKIE['cookie_index_sourcelang']))
{

$source_index =$_COOKIE['cookie_index_sourcelang'] ;
    $_SESSION['source_index'] = $_COOKIE['cookie_index_sourcelang'];

}
```

**Gestione cookie**

**(fig.5)**

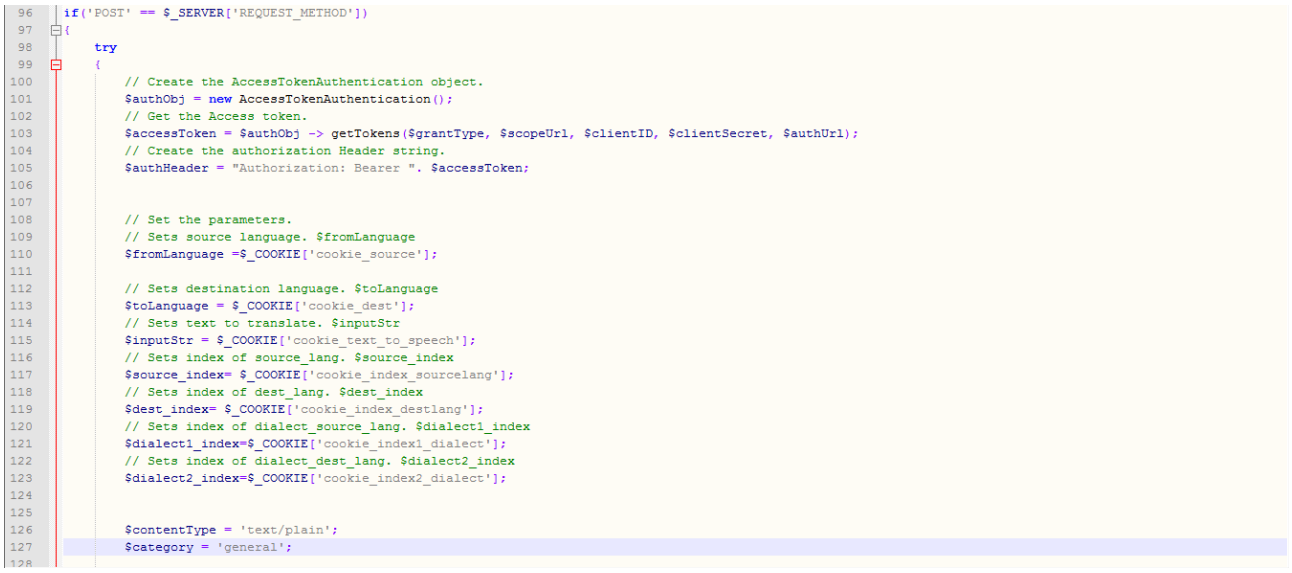
```php
    $_SESSION['dest_index'] = $_COOKIE['cookie_index_destlang'];

}

// Sets session variable for index of source dialect
if (isset($_COOKIE['cookie_index1_dialect']))
{

$dialect1_index =$_COOKIE['cookie_index1_dialect'] ;
    $_SESSION['dialect1_index'] = $_COOKIE['cookie_index1_dialect'];


}

// Sets session variable for index of destination dialect
if (isset($_COOKIE['cookie_index2_dialect']))
{

$dialect2_index =$_COOKIE['cookie_index2_dialect'] ;
    $_SESSION['dialect2_index'] = $_COOKIE['cookie_index2_dialect'];

}
```

Il ripristino di una sessione creata in precedenza avverrà sempre con il comando session_start() e ciò consentirà di richiamare variabili di sessione precedentemente valorizzate (leggendo i dati che il server ha salvato sul file di sessione e rendendoli disponibili nello script php corrente). Per ripristino si intende, quindi, la verifica dell'esistenza del cookie SID sul pc dell'utente: una

volta individuato per il server sarà possibile risalire al suo file di sessione e, quindi, alle variabili salvate al suo interno.

In tal modo si assicura la persistenza delle informazioni impostate alla prima esecuzione dell'applicazione. Il sito verrà richiamato dopo la traduzione, ma manterrà i dati scelti o generati durante la fase di riconoscimento e settaggio delle lingue sorgenti e di destinazione per effettuare la traduzione.

Si procede dunque con la traduzione del testo riconosciuto. Racchiuso all'interno del blocco try-catch, utile alla gestione di eventuali errori, si ottiene, tramite un token, il permesso ad utilizzare le API Microsoft. La traduzione avviene utilizzando il metodo 'Translate', invocato attraverso l'URL creato precedentemente. Una volta ottenuta la traduzione, in formato XML, esso viene interpretato in modo che diventi un oggetto che possa essere usato nell'applicazione. Il risultato della traduzione viene salvato nella variabile $translatedText.



```
96  if('POST' == $_SERVER['REQUEST_METHOD'])
97  {
98      try
99      {
100         // Create the AccessTokenAuthentication object.
101         $authObj = new AccessTokenAuthentication();
102         // Get the Access token.
103         $accessToken = $authObj -> getTokens($grantType, $scopeUrl, $clientID, $clientSecret, $authUrl);
104         // Create the authorization Header string.
105         $authHeader = "Authorization: Bearer ". $accessToken;
106
107
108         // Set the parameters.
109         // Sets source language. $fromLanguage
110         $fromLanguage =$_COOKIE['cookie_source'];
111
112         // Sets destination language. $toLanguage
113         $toLanguage = $_COOKIE['cookie_dest'];
114         // Sets text to translate. $inputStr
115         $inputStr = $_COOKIE['cookie_text_to_speech'];
116         // Sets index of source_lang. $source_index
117         $source_index= $_COOKIE['cookie_index_sourcelang'];
118         // Sets index of dest_lang. $dest_index
119         $dest_index= $_COOKIE['cookie_index_destlang'];
120         // Sets index of dialect_source_lang. $dialect1_index
121         $dialect1_index=$_COOKIE['cookie_index1_dialect'];
122         // Sets index of dialect_dest_lang. $dialect2_index
123         $dialect2_index=$_COOKIE['cookie_index2_dialect'];
124
125
126         $contentType = 'text/plain';
127         $category = 'general';
128
```

**Assegnamento cookie a variabile**

**(fig.6)**

```php

    // Variable that composes the string of parameters for the transaltion
    $paramst = "text=".urlencode($inputStr)."&to=".$toLanguage."&from=".$fromLanguage;
    // URL to translate the text
    $translateUrl = "http://api.microsofttranslator.com/v2/Http.svc/Translate?$paramst";

    //Create the Translator Object.
    $translatorObj = new HTTPTranslator();

    //Get the curlResponse.
    $curlResponse = $translatorObj -> curlRequest($translateUrl, $authHeader);

    //Interprets a string of XML into an object.
    $xmlObj = simplexml_load_string($curlResponse);
    foreach((array)$xmlObj[0] as $val)
    {
        $translatedStr = $val;
    }


    $translatedText = urlencode($translatedStr);


```

**Richiesta traduzione**

**(fig.7)**

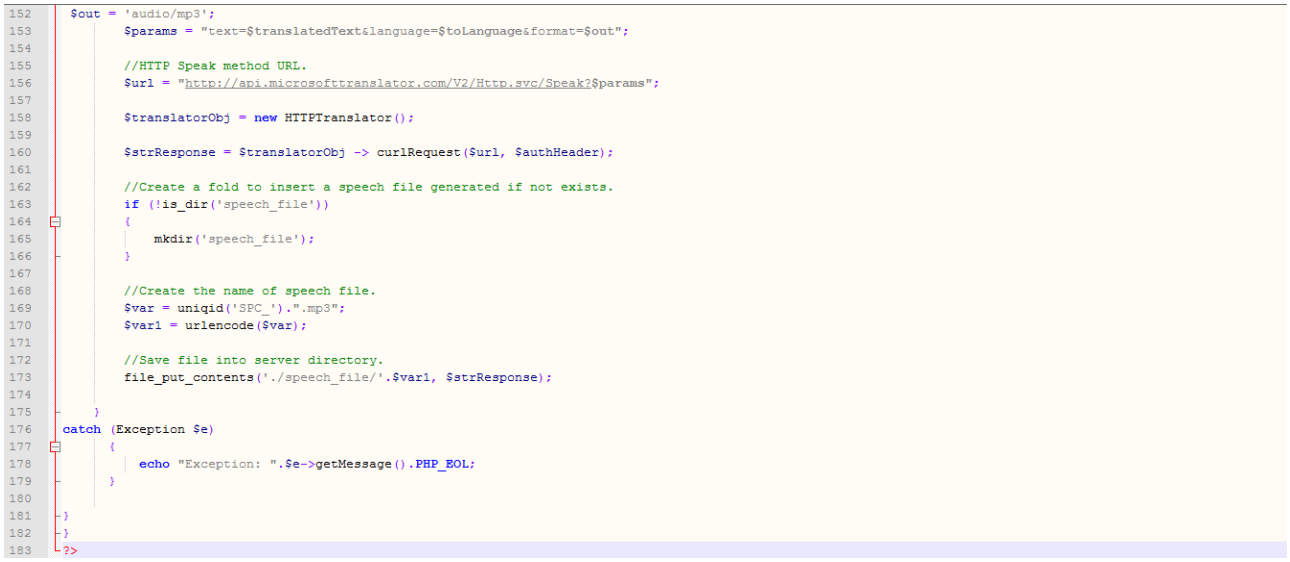
Dopo aver ottenuto il testo tradotto, viene anche generato il text-to-speech. Anche qui si vanno a settare delle variabili così che si possa costruire l'URL di richiesta. Il metodo invocato per ottenere il text-to-speech è lo 'Speak' che richiede il settaggio dei seguenti parametri:

- il testo tradotto;

- la lingua in cui il testo è stato tradotto;

- il formato in cui restituire lo stream audio.

Per quanto riguarda il formato dello stream audio, nell'implementazione è stato scelto '.mp3'.



```
$out = 'audio/mp3';
        $params = "text=$translatedText&language=$toLanguage&format=$out";

        //HTTP Speak method URL.
        $url = "http://api.microsofttranslator.com/V2/Http.svc/Speak?$params";

        $translatorObj = new HTTPTranslator();

        $strResponse = $translatorObj -> curlRequest($url, $authHeader);

        //Create a fold to insert a speech file generated if not exists.
        if (!is_dir('speech_file'))
        {
            mkdir('speech_file');
        }

        //Create the name of speech file.
        $var = uniqid('SPC_').".mp3";
        $var1 = urlencode($var);

        //Save file into server directory.
        file_put_contents('./speech_file/'.$var1, $strResponse);

    }
catch (Exception $e)
    {
        echo "Exception: ".$e->getMessage().PHP_EOL;
    }

}
}
?>
```

**Creazione speech-file**

**(fig.8)**

Una volta ottenuto lo stream audio, esso viene memorizzato nella cartella ‘speech_file’, la quale viene creata nel caso in cui non fosse già presente. Al file viene assegnato ogni volta un nome univoco, ottenuto grazie all’istruzione ‘uniqid('SPC_').".mp3" ‘, e viene salvato nella directory utilizzando la funzione *‘file_put_contents('./speech_file/'.$var, $strResponse)’* dove il primo argomento indica la cartella dove salvare il file, mentre il secondo argomento indica il file.

Salvato il file audio, viene chiuso il try-catch, ponendo fine allo script PHP.

## 5.4.2 Javascript

Il codice utilizza le web speech API di Google per il riconoscimento vocale e comprende le seguenti definizioni di bottoni, variabili e funzioni:

Definizione dei riferimenti dalle quali poter accedere, questa volta senza autenticazione, alle API Google per la traduzione. Negli href si fa riferimento a tutti gli indirizzi dove sono presenti i contenuti utilizzati dalle API di Google :

```
<!DOCTYPE html>
<html class="no-js consumer" lang="en">
  <head>
  <link rel="stylesheet" href="css.css" type="text/css">
    <script>
(function(e, p){
    var m = location.href.match(/platform=(win8|win|mac|linux|cros)/);
    e.id = (m && m[1]) ||
          (p.indexOf('Windows NT 6.2') > -1 ? 'win8' : p.indexOf('Windows') > -1 ? 'win' : p.indexOf('Mac') > -1 ? 'mac' : p.indexOf('CrOS'
    e.className = e.className.replace(/\bno-js\b/,'js');
  })(document.documentElement, window.navigator.userAgent)
    </script>
    <meta charset="utf-8">
    <meta content="initial-scale=1, minimum-scale=1, width=device-width" name="viewport">
    <meta content=
    "Google Chrome is a browser that combines a minimal design with sophisticated technology to make the web faster, safer, and easier."
    name="description">
    <title>
     Progetto Speech Recognition and Translation
    </title>
    <link href="https://plus.google.com/100585555255542998765" rel="publisher">
    <link href="//www.google.com/images/icons/product/chrome-32.png" rel="icon" type="image/ico">
    <link href="//fonts.googleapis.com/css?family=Open+Sans:300,400,600,700&amp;subset=latin" rel=
    "stylesheet">

    <script src="//www.google.com/js/gweb/analytics/autotrack.js">
</script>
```

Definizione e controllo degli errori che possono sorgere al click del pulsante del riconoscimento vocale (rappresentato graficamente da un microfono).

Essendo il microfono un componente non sempre presente o utilizzato nel PC, viene richiesto un consenso da parte dell'utente all'utilizzo del microfono che attiva le funzionalità dello stesso e per permetterne l'utilizzo all'interno del web browser.

Tuttavia è possibile che si incorra in errori dovuti al mal funzionamento dell'hardware. Dunque, vengono gestite le eventuali problematiche legate all'utilizzo del microfono con dei messaggi di errore che esplicitano il bug sollevato e le informazioni per l'utilizzo iniziale del microfono. Ad ogni errore di impostazione del microfono è associato un messaggio e un riferimento ad una guida help fornita da Chrome.

```
</head>
<body class="" id="grid">
  <div class="browser-landing" id="main">
    <div class="compact marquee-stacked" id="marquee">
      <div id = "header">

      </div>
      </div>
    </div>
    <div class="compact marquee">
      <div id="info">

        <p id="info_no_speech" style="display:none">

     Nessun riconoscimento vocale è stato trovato. Controlla le impostazioni del microfono<a href=
          "//support.google.com/chrome/bin/answer.py?hl=en&amp;answer=1407892">microphone settings</a>.

        </p>
        <p id="info_no_microphone" style="display:none">

      Nessun microfono è stato trovato. Assicurati che sia installato<a href="//support.google.com/chrome/bin/answer.py?hl=en&amp;answer=
          microphone settings</a> are configured correctly.

        </p>
        <p id="info_allow" style="display:none">

      Clicca "Consenti" per abilitare il microfono.
        </p>
        <p id="info_denied" style="display:none">
```

In questa sezione vengono istanziati anche:

- Il bottone per avviare il riconoscimento vocale “start_button” che è rappresentato graficamente da un microfono “mic.gif” il cui click richiama la funzione startButton(event) che aziona il riconoscimento vocale.
- Il bottone che rappresenta il feedback di riconoscimento in corso “animate”, rappresentato da un microfono lampeggiante il cui clik richiama la funzione “stop_rec()” che termina il riconoscimento vocale.
- Le due textarea (“results”, “results2”) che conterranno rispettivamente il testo riconosciuto e il testo tradotto. Si effettua un controllo sulle variabili php $inputStr e $translatedStr per il riempimento delle aree con il testo riconosciuto e quello tradotto. Nel caso in cui sino già avvenuti il riconoscimento e/o la traduzione, le due aree conterranno rispettivamente il testo riconosciuto e tradotto.

```
250         </p>
251         <p id="info_upgrade" style="display:none">
252       Web Speech Api non è supportato da questo browser. Effettua l'upgrade<a href=
253           "//www.google.com/chrome">Chrome</a> alla versione 25 o dopo.
254         </p>
255       </div>
256       <div id="container">
257       <div id="div_start">
258
259         <button id="start_button" class="start_button" type="button" onclick="startButton(event)"></button>
260       </div>
261       <div id="animato">
262
263       <button id="animate" class="animate" type="button" style="display:none" onclick="stop_rec()" >
264       </button>
265       </div>
266       <div id="results">
267         <span class="final" id="final_span"></span> <span class="interim" id=
268         "interim_span"></span>
269          <?php if (isset($inputStr)== true){echo $inputStr;}else{echo '';} ?>
270       </div>
271
272       <div id="results2">
273        <span class="final" id="final_span2"></span> </span>
274         <?php if (isset($inputStr)== true){echo $translatedStr;}else{echo '';} ?>
275       </div>
276
277
278
```

**Definizione bottoni e textarea**

**(fig.11)**

Successivamente si definiscono delle variabili globali e degli elementi che permettono il corretto utilizzo della API Google speech recognition. Inoltre si avvalora la lingua di default utilizzata dall'API.

```
<script>
var source='';
var dest='';
var final_transcript='';

window.___gcfg = { lang: 'en' };
 (function() {
   var po = document.createElement('script'); po.type = 'text/javascript'; po.async = true;
   po.src = 'https://apis.google.com/js/plusone.js';
   var s = document.getElementsByTagName('script')[0]; s.parentNode.insertBefore(po, s);
 })();




   </script> <script>
var langs =
[


 ['German',         ['de']],
 ['English',         ['en', 'Australia'],
                     ['en', 'Canada'],
                     ['en', 'India'],
```

**Caricamento lingua di default**

**(fig.12)**

Si procede con la definizione di tutte le lingue disponibili per il riconoscimento. Nell'array, il primo campo è quello visualizzato nel combo box, e indica la lingua desiderata, il secondo, rappresenta l'acronimo che verrà successivamente utilizzato per tradurre lo speech riconosciuto e l'ultimo che rappresenta i dialetti presenti e riconoscibili vocalmente per ogni lingua definita.

**Definizione lingue**

**(fig.13)**

```
                    ['en', 'South Africa'],
                    ['en', 'United Kingdom'],
                    ['en', 'United States']],
 ['Español',        ['es', 'Argentina'],
                    ['es', 'Bolivia'],
                    ['es', 'Chile'],
                    ['es', 'Colombia'],
                    ['es', 'Costa Rica'],
                    ['es', 'Ecuador'],
                    ['es', 'El Salvador'],
                    ['es', 'España'],
                    ['es', 'Estados Unidos'],
                    ['es', 'Guatemala'],
                    ['es', 'Honduras'],
                    ['es', 'México'],
                    ['es', 'Nicaragua'],
                    ['es', 'Panamá'],
                    ['es', 'Paraguay'],
                    ['es', 'Perú'],
                    ['es', 'Puerto Rico'],
                    ['es', 'República Dominicana'],
                    ['es', 'Uruguay'],
                    ['es', 'Venezuela']],

 ['Français',       ['fr']],

 ['Italiano',       ['it', 'Italia'],
                    ['it', 'Svizzera']],

 ['Nederlands',     ['nl']],

 ['Polski',         ['pl']],
 ['Português',      ['pt', 'Brasil'],
                    ['pt', 'Portugal']],
 ['Română',         ['ro']],

 ['Svenska',        ['sv']],
 ['Türkçe',         ['tr']],

 ['Русский',        ['ru']],


['한국어',          ['ko']],
 ['中文',           ['zh', '普通话 (中国大陆)'],
                    ['zh', '普通话 (香港)'],
                    ['zh', '中文 (台灣)'],
                    ['zh', '粵語 (香港)']],
 ['日本語',         ['ja']]];

for (var i = 0; i < langs.length; i++) {
  select_language.options[i] = new Option(langs[i][0], i);
select_language2.options[i] = new Option(langs[i][0], i);
}
verify();

updateCountry();
updateCountry2();
```

La funzione verify(), verifica che non sia stata selezionata alcuna lingua precedentemente. Se vero implica che è in esecuzione la fase di traduzione del testo e quindi i valori all'interno delle combo box contenenti le lingue, vengono reimpostati alla lingua scelta per il riconoscimento e successiva traduzione.

Altrimenti si avvalorano i campi alle lingue di default Italiano- Inglese.

```
showInfo('info_start');
var translated_string;

//evalue the list box with the selected language and dialect, otherwise whit a defalult index
function verify()
{<?php if (isset($source_index) == true)
                    {
                    ?>
                    var source_index = "<?php echo $source_index; ?>";
                    var dest_index= "<?php echo $dest_index; ?>";

                    select_language.selectedIndex = source_index;
                    select_language2.selectedIndex = dest_index;

                    <?php
                    } else {?>
                    select_language.selectedIndex = 4;
                    select_language2.selectedIndex = 1;
                    select_dialect.selectedIndex = 0;
                    select_dialect2.selectedIndex = 6;
                    <?php }?>
                    }

```

**Imposta lingua combo_box**

**(fig.14)**

Nel seguito si avvalorano le combo box con le lingue e si richiamano le funzioni updateCountry() e updateCountry2(). Esse, a seconda delle lingua sorgente/destinazione selezionata, avvalorano i rispettivi campi dialettali.

```

  <!-- Depending of the source language choosen, evalue the corresponding dialect-->
function updateCountry() {

 for (var i = select_dialect.options.length - 1; i >= 0; i--) {
   select_dialect.remove(i);
 }
 var list = langs[select_language.selectedIndex];
 for (var i = 1; i < list.length; i++) {
   select_dialect.options.add(new Option(list[i][1], list[i][0]));
 }
 <?php if (isset($dialect1_index) == true)
                    {
                                ?>
                                var dialect1_index= "<?php echo $dialect1_index; ?>";
                                select_dialect.selectedIndex=dialect1_index;
            <?php } else {?>
 <?php }?>
 select_dialect.style.visibility = list[1].length == 1 ? 'hidden' : 'visible';
}


  <!-- Depending of the destination language choosen, evalue the corresponding dialect-->
function updateCountry2() {
  for (var i = select_dialect2.options.length - 1; i >= 0; i--) {
    select_dialect2.remove(i);
  }
  var list = langs[select_language2.selectedIndex];
  for (var i = 1; i < list.length; i++) {
    select_dialect2.options.add(new Option(list[i][1], list[i][0]));
  }

```

**Riempimento combo box**

**(fig.15)**

Si inizializza, ora, la variabile “recognition” per il riconoscimento vocale creando l’oggetto webkitSpeechRecognition(). Si imposta a vero l’attivazione

del riconoscimento vocale con la proprietà “.continuous”. La proprietà “.interimResults” definisce il risultato intermedio del riconoscimento che viene quindi attivato. Nel momento in cui viene dato avvio al riconoscimento, si richiama la funzione “function()” assegnata alla proprietà “.onstart”. Nella funzione viene inizializzato il riconoscimento e graficamente si nota il microfono lampeggiante.

```
var recognizing = false;
var ignore_onend;
var start_timestamp;
var current_style;

if (!('webkitSpeechRecognition' in window)) {
  upgrade();
} else {
  start_button.style.display = 'inline-block';
  buttons.style.display = 'none';

  var recognition = new webkitSpeechRecognition();
  recognition.continuous = true;
  recognition.interimResults = true;
  recognition.onstart = function() {
    recognizing = true;
    start_button.style.display = 'none';
    animate.style.display = 'inline-block';


  };
```

**Avvio riconoscimento vocale**

**(fig.16)**

Si richiamano le funzioni che gestiscono gli eventi che possono causare errori nel riconoscimento vocale che vengono dunque, gestiti tramite “.onerror” e “.onend”. Se un errore dovesse essere riconosciuto dalla proprietà onerror, si tratta di errori del tipo: non è stato riconosciuto alcuno speech, non è stato riconosciuto alcun microfono, l’hardware microfono è bloccato.

Con il controllo della proprietà “.onend”, si termina il riconoscimento vocale.

```
487
488   };
489   recognition.onerror = function(event) {
490     if (event.error == 'no-speech') {
491       start_img.src = 'image/mic.gif';
492       showInfo('info_no_speech');
493       ignore_onend = true;
494     }
495     if (event.error == 'audio-capture') {
496       start_img.src = 'image/mic.gif';
497       showInfo('info_no_microphone');
498       ignore_onend = true;
499     }
500     if (event.error == 'not-allowed') {
501       if (event.timeStamp - start_timestamp < 100) {
502         showInfo('info_blocked');
503       } else {
504         showInfo('info_denied');
505       }
506       ignore_onend = true;
507     }
508   };
509
510   recognition.onend = function() {
511     recognizing = false;
512     if (ignore_onend) {
513       return;
514     }
515     start_img.src = 'image/mic.png';
```

**Gestione errori**

**(fig.17)**

Di seguito viene definito il codice che effettua il riconoscimento. Vengono dunque avvalorate le varabili interim_transcript e final_transcript che conterranno la stringa del riconoscimento vocale. Nella funzione viene gestito il caso in cui non venga riconosciuto alcuno speech. In questo caso, il riconoscimento viene stoppato e l'applicazione viene riportata alla situazione iniziale.

Altrimenti, il sistema è in fase di riconoscimento del parlato.

Se "event.result[i].isFinal" è vera, è stata riconosciuta una stringa come "finale" e quindi non modificabile. Viene, perciò, avvalorata la variabile final_transcript, accodando al precedente contenuto di essa, il nuovo valore riconosciuto.

In caso contrario, viene avvalorata la variabile interim_transcript che rappresenta il riconoscimento intermedio del parlato.

Questo procedimento viene ripetuto in un for fino a che vengono riconosciute dal sistema delle parole. Il for, infatti, termina quando la lunghezza del risultato ottenuto dal riconoscimento non viene raggiunta.

Infine, viene impostata la prima lettera della stringa riconosciuta come finale, in maiuscolo.

```
    showInfo('');
    if (window.getSelection) {
      window.getSelection().removeAllRanges();
      var range = document.createRange();
      range.selectNode(document.getElementById('final_span'));
      window.getSelection().addRange(range);
    }

  };
  recognition.onresult = function(event) {
    var interim_transcript = '';
    if (typeof(event.results) == 'undefined') {
      recognition.onend = null;
      recognition.stop();
      upgrade();
      return;
    }
    for (var i = event.resultIndex; i < event.results.length; ++i) {
      if (event.results[i].isFinal) {
        final_transcript += event.results[i][0].transcript;
      } else {
        interim_transcript += event.results[i][0].transcript;
      }
    }
    final_transcript = capitalize(final_transcript);

    final_span.innerHTML = linebreak(final_transcript);
    interim_span.innerHTML = linebreak(interim_transcript);
    if (final_transcript || interim_transcript) {
      showButtons('inline-block');
    }
  };
```

**Avvaloramento final_transcript**

**(fig.18)**

La funzione Translate(), è una delle funzioni chiave dell’applicazione. Se il riconoscimento va a buon fine, setta tutte le variabili che verranno passate al server e che verranno utilizzate dalle API di Microsoft Translate per la traduzione. In particolare verranno passati tutti gli indici dell’array delle lingue e dei dialetti (per mantenere vive le variabili settate dopo il refresh della pagina) e l’acronimo di identificazione della lingua sorgente e di quella di destinazione che verrà passato al traduttore.

Per i passaggi delle variabili al server sono stati utilizzati i cookie in quanto rispondono alla necessità di memorizzare in modo più o meno permanentemente alcuni dati e informazioni durante la navigazione

dell'utente nel sito in modo da essere riutilizzati successivamente in altre pagine**.**

Nel nostro caso specifico, abbiamo necessità di mantenere persistenti le informazioni delle lingue, dei rispettivi dialetti e del testo riconosciuto dallo speech quando si passa il controllo al server per la traduzione.

La pagina, durante la traduzione, viene ricaricata ma i dati precedentemente settati devono restare tali.

I cookie sono il metodo per memorizzare, sul computer dell'utente, delle informazioni che vogliamo persistano anche nelle successive visite al nostro sito.

Nella nostra applicazione vengono settati con il comando
*document.cookie= 'nome cookie='+variabile;*
la variabile è il valore che vogliamo persista all'interno del sito nelle esecuzioni successive.

Infine, viene effettuato un controllo: se la variabile $translatedStr è avvalorata (il che implica che la traduzione è stata già fatta dal server), allora viene visualizzata nella seconda result area.

```
function Translate() {
  if (recognizing) {
     recognizing = false;
     recognition.stop();
//set variable for the trasmission to the server
text_to_speech=final_transcript;
index_sourcelang= select_language.selectedIndex;
index_destlang= select_language2.selectedIndex;
index1_dialect= select_dialect.selectedIndex;
index2_dialect=select_dialect2.selectedIndex;
document.cookie='cookie_index2_dialect='+index2_dialect;
document.cookie= 'cookie_index_sourcelang='+index_sourcelang;
document.cookie= 'cookie_index_destlang='+index_destlang;
document.cookie= 'cookie_text_to_speech='+text_to_speech;
document.cookie= 'cookie_index1_dialect='+index1_dialect;
source=langs[select_language.selectedIndex][1][0];
document.cookie= 'cookie_source='+source;
dest=langs[select_language2.selectedIndex][1][0];
document.cookie= 'cookie_dest='+dest;

final_span.innerHTML=linebreak(final_transcript);

buttons.style.display = style;

translated_string='<?php if (isset($translatedStr)== true){echo $translatedStr;}else{echo '';} ?>';

    }
else{
}

  copy_info.style.display = 'inline-block';
  showInfo('');
```

**Passaggio al server dei variabili attraverso i cookie**

**(fig.19)**

La funzione startButton(event) viene richiamata al click sul microfono. Viene inizializzato quindi il riconoscimento con “recognition.start()”. Se non vengono consentiti i permessi per l’utilizzo del microfono, il riconoscimento non può aver inizio.

```
function startButton(event) {
  if (recognizing) {
    recognition.stop();
    return;
  }
  recognition.lang = select_dialect.value;
  recognition.start();
  ignore_onend = false;
  final_span.innerHTML = '';
  interim_span.innerHTML = '';
  start_img.src = 'mic-slash.gif';
  showInfo('info_allow');
  showButtons('none');
  start_timestamp = event.timeStamp;
}

function upgrade() {
  start_button.style.visibility = 'hidden';
  showInfo('info_upgrade');
}
var two_line = /\n\n/g;
var one_line = /\n/g;
function linebreak(s) {
  return s.replace(two_line, '<p></p>').replace(one_line, '<br>');
}
var first_char = /\S/;
function capitalize(s) {
  return s.replace(first_char, function(m) { return m.toUpperCase(); });
}
```

**Start riconoscimento**

**(fig.20)**

Le funzioni showInfo() e showButtons(style) hanno il compito semplicemente di mostrare informazioni e bottoni che durante la fase che precede la traduzioni sono resi invisibili.

```
function showInfo(s) {
  if (s) {
    for (var child = info.firstChild; child; child = child.nextSibling) {
      if (child.style) {
        child.style.display = child.id == s ? 'inline' : 'none';
      }
    }
    info.style.visibility = 'visible';
  } else {
    info.style.visibility = 'hidden';
  }
}


function showButtons(style) {
  if (style == current_style) {
    return;
  }
  current_style = style;
  buttons.style.display = style;

  copy_button.style.display = style;

  copy_info.style.display = 'none';

}
```

**Mostrare info e bottoni**

**(fig.21)**

Si implementa infine la funzione per lo speech audio per ascoltare con voce “elettronica”, la pronuncia del testo tradotto nella lingua desiderata:

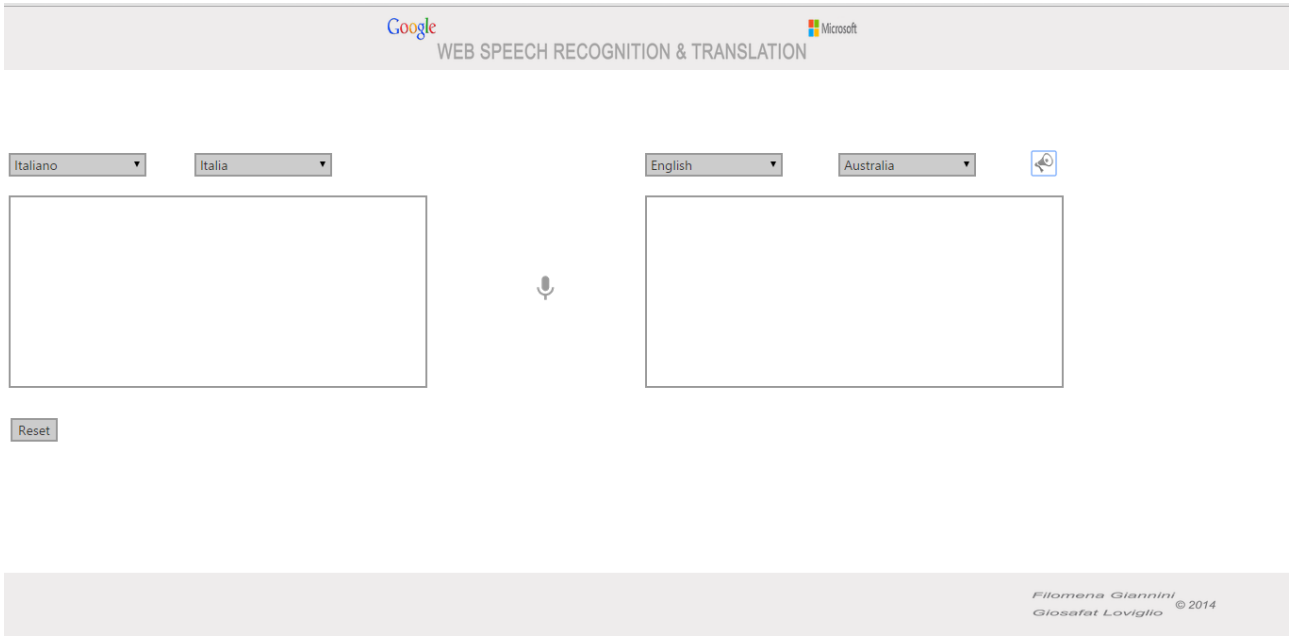
```
function speech_play()
{

    var play_speech = document.getElementById("play_speech");
    if (play_speech.paused)
    {
        play_speech.play();
    }
    else
    {
        play_speech.pause();
    }
}
```

**Speech player**

**(fig.22)**

# 6. L’applicazione in esecuzione

Web Speech Recognition Translator si presenta con un’interfaccia minimale quanto efficace, intuitiva e usabile. Al lancio dell’applicazione vengono caricati tutti i contenuti e avvalorate le variabili opportune. (da fig.9 a fig.15)




Si fornisce di seguito una panoramica di come si comporta, in esecuzione, l’applicazione.

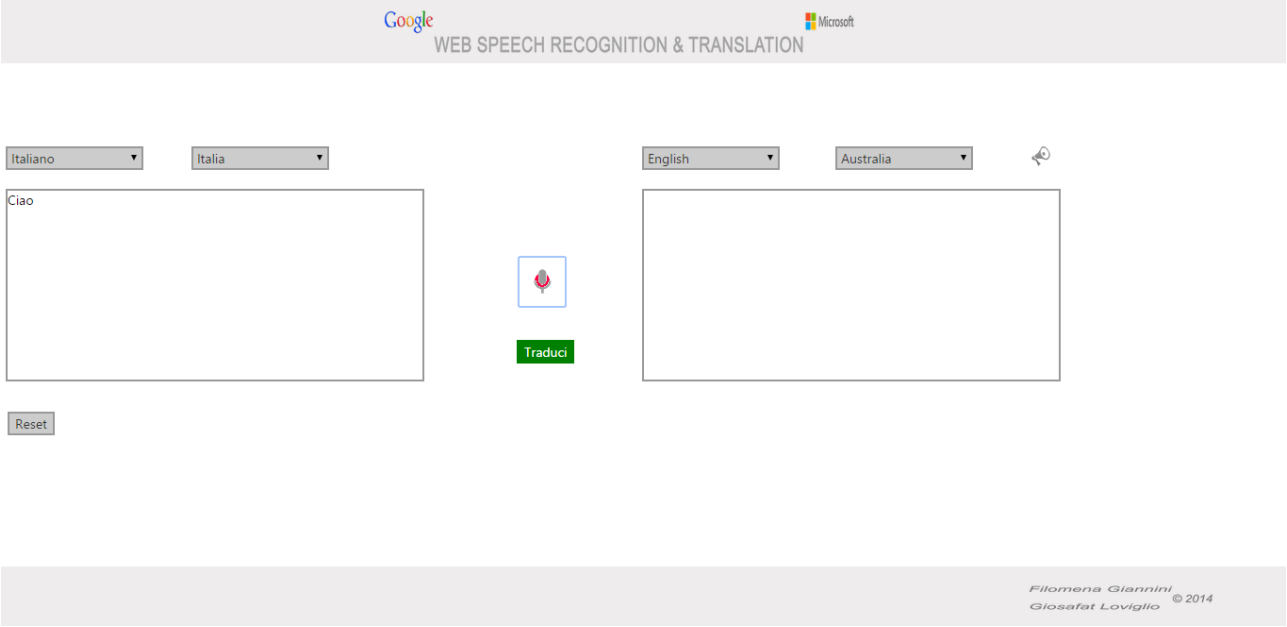
## 6.1 Client side

Per cominciare ad utilizzare l'applicazione, è necessario cliccare sullo start_button (graficamente rappresentato con l'icona del microfono) che richiamerà la funzione startButton(event).

La funzione, che utilizza le API di Google web speech recognition, riconosce il parlato, inizializza le variabili di riconoscimento intermedio e riconoscimento definitivo dello speech.

Viene eseguito il main code, dove si avvalorano le variabili interim_transcript e final_transcript (fig.16-17-18). L'ascolto da parte dell'applicazione continua fino a quando si verifica uno dei seguenti casi:

- Al click dello start_button (microfono), che stoppa il riconoscimento.
- Non viene riconosciuta nessuna parola quindi si reimposta la situazione iniziale.
- Al click del bottone "Traduci" che avvia la traduzione delle parole riconosciute dall'applicazione e che appare dopo il primo riconoscimento effettivo.




Per riconoscimento effettivo intendiamo l'avvaloramento della variabile final_transcritpt che graficamente è espressa dalla stringa di colore nero. Essa non viene più modificata né influenzata dalle parole pronunciate successivamente né dal loro contesto.

Come detto precedentemente, invece, la variabile interim_transcript rappresenta una sorta di "loading" del parlato. La stringa qui riconosciuta appare in grigio chiaro e può modificarsi a seconda dei riconoscimenti immediatamente successivi ad essa per contestualizzare la parola. Il riconoscitore, quando ritiene di aver individuato la parola esatta, passa il contenuto di essa alla variabile final_transcript.

L'output viene visualizzato nella "results" area.

Al click del bottone "Traduci" (button_translate), vengono attivate le API di traduzione con la funzione "Translate()" che tradurrà la stringa riconosciuta in final_transcript.(fig.19)

Questa funzione setta tutte le variabili che ci serviranno nel php per effettuare la traduzione. Si passano infatti, i valori di indice di tutte le lingue e dialetti, e gli acronimi delle lingue che sono utilizzate nelle API di Microsoft Translate

per tradurre le stringe dalla lingua sorgente a quella di destinazione. Il passaggio client-server viene effettuato attraverso i cookie.

## 6.2 Server side

A questo punto, vengono ricevute dal server, tutte le variabili necessarie e procedere con la traduzione.(fig.4-5-6-7)

Avvalorando le variabili:

$paramst =“text=".urlencode($inputStr)."&to=".$toLanguage."&from=".$fromLanguage;

$translateUrl= $translateUrl = "http://api.microsofttranslator.com/v2/Http.svc/Translate?$paramst";

viene effettuata la traduzione del testo riconosciuto precedentemente.

L’output della traduzione è visualizzato e gestito nella funzione javascript “Translate()” con la variabile

translated_string='<?php if (isset($translatedStr)== true){echo $translatedStr;}else{echo '';} ?>';.

Se, dopo la traduzione, si vuole ascoltare la pronuncia, è possibile cliccare sull'icona del megafono (t2s).
Viene richiamata, quindi, la funzione speech_play(). Questa funzione, sempre grazie alle API di Microsoft Translate, aziona una voce elettronica che legge, nella lingua di destinazione, l'output della traduzione.(fig.22)

L'applicazione, dunque, ha terminato il suo compito.
Si è acquisito il parlato, effettuata la traduzione ed ascoltata la pronuncia del testo tradotto.
A questo punto, è possibile cliccare sul bottone "Reset", per effettuare una nuova acquisizione del vocale e rispettiva traduzione, o chiudere l'applicazione.
Il click sul bottone "reset", azzera i valori delle result box. È possibile, quindi, procedere con un nuovo riconoscimento.

# 7. Conclusioni e sviluppi futuri

Integrando diverse tecnologie e partendo da esigenze di carattere comune e quanto più evidenziate dalla necessità di collaborare in ambienti applicativi e lavorativi con persone di diverse nazionalità, si è sviluppata un'applicazione di web speech recognition e traduzione.

Essa, può trovare larga applicazione in conversazioni vocali nelle quali è necessario avere un feedback, in tempi ridotti, di un discorso o di una frase pronunciata in una lingua straniera. O anche, può essere di supporto all'utilizzatore nella traduzione di discorsi che, se scritti, necessiterebbero di un tempo ben più lungo di quello impiegato nel parlato.

Evitare errori ortografici può anche essere facilitato da questo tipo di applicazione che non richiede nessun input testuale da parte dell'utente.

Possiamo quindi affermare che la Web Application Speech Recognition and Translation, può riscontrare nella vita reale molteplici concrete funzionalità.

La comunicazione client server avviene in maniera semplice e con tempi moderati.

Si potrebbe pensare nel futuro, di rendere la traduzione contemporanea al riconoscimento parlato e l'inserimento di ulteriori lingue (in concomitanza con le possibilità offerte dalle API utilizzate).

# 8. Bibliografia


- https://github.com/collab-uniba/speech-translation-tools/tree/microsoft-translator-api
- https://www.google.com/intl/it/chrome/demos/speech.html
- http://www.microsoft.com/en-us/translator/developers.aspx
- https://stackoverflow.com/
- http://antirez.com/latest/0
- http://www.html.it/pag/16696/mantenere-lo-stato-i-cookie/
- http://updates.html5rocks.com/2013/01/Voice-Driven-Web-Apps-Introduction-to-the-Web-Speech-API