

Implementazione di un traduttore utilizzando le

‘Microsoft Translator API’

(Microsoft Translator)

Sistemi per la Collaborazione in rete – Prof. Filippo Lanubile, Fabio Calefato

Salvatore De Astis

13-14

Sommario

[Introduzione 2](#_Toc391376878)

[Implementazione 3](#_Toc391376879)

[Creazione dell’account Microsoft 3](#_Toc391376880)

[Creazione dell’account sul Marketplace di Windows Azure 3](#_Toc391376881)

[Implementazione dell’applicazione ‘Microsoft Traduttore’ 11](#_Toc391376882)

[Modulo ‘credential.php’ 11](#_Toc391376883)

[Modulo ‘AccessTokenAuthentication.php’ 12](#_Toc391376884)

[Modulo ‘HttpTranslator.php’ 13](#_Toc391376885)

[Modulo ‘translate.php’ 14](#_Toc391376886)

[Modulo ’functions\_js.js’ 19](#_Toc391376887)

[Modulo ‘css.css’ 19](#_Toc391376888)

[Installazione dell’applicazione in XAMPP 22](#_Toc391376889)

# Introduzione

Mentre il mondo diventa sempre più piccolo grazie all’utilizzo di molti strumenti di comunicazione come smartphone, tablet, personal computer, ecc., l’unico ostacolo che persiste tra persone di culture differenti è la lingua, e proprio per superare quest’ultima barriera di comunicazione che, in ambito applicativo, si stanno diffondendo i traduttori automatici.

Infatti per rendere le pagine web o le applicazioni utilizzabili anche da utenti con nazionalità e cultura differente da quella dello sviluppatore, vengono utilizzati traduttori automatici, i quali rimangono uno dei modi più efficaci per colmare il divario imposto dalla lingua. Ma non sempre la miglior tecnologia di traduzione automatica fornisce traduzioni accurate o adatte per un sito o per l’utente, proprio come accade con un traduttore umano. Inoltre, molte pagine web non sono totalmente tradotte, ma presentano solamente delle sezioni che sono disponibili in varie lingue; questo accade per colpa dell’elevato costo della traduzione e del costo che la traduzione ha nel tempo, dato dall’eventuali aggiornamenti del sito. Lo stesso vale per le applicazioni; è impossibile pensare che uno sviluppatore si affidi a dei traduttori umani così da rendere i contenuti dell’applicazione disponibili in varie lingue. Proprio in questo contesto che entra in gioco Microsoft Translator.

Nello specifico, Microsoft Translator è un servizio host accessibile via API (Application Program Interface), che permette di tradurre le applicazioni senza che sia necessario un traduttore umano. Può essere utilizzato per utilizzato in qualsiasi scenario in cui c’è bisogno di una traduzione, dalla reazione di una applicazione per smartphone o tablet, alla creazione di applicativi desktop senza dimenticarci di pagine web ecc. La traduzione può essere fatta utilizzando il widget di Microsoft Translator o un API Ajax, che possono essere integrati perfettamente nel sito senza il bisogno di ulteriori sforzi. In più il widget ha la capacità di migliorare la l’accuratezza grammaticale e contestuale delle traduzioni. Tutto il lavoro di traduzione viene effettuato grazie ai metodi *AddTranslation()* e *GetTranslations()* che formano, insieme anche al metodo *Translate()*, il **Collaborative Translation Framework**.

La traduzione non viene fatta solo attraverso un widget, ma anche attraverso varie interfacce come quella SOAP, http e Ajax. Il progetto per il corso di Sistemi per la collaborazione in rete si focalizza nel costruire un traduttore che utilizzi le API utilizzando l’interfaccia HTTP.

La seguente documentazione andrà a spiegare il funzionamento delle API attraverso il protocollo HTTP, andando a mostrare tutte le fasi necessarie per implementare il traduttore, senza tralasciare le fasi preliminari che servono per ottenere le credenziali token, necessarie per poter utilizzare la API.

# 

# Implementazione

Di seguito verrà mostrato lo sviluppo del traduttore. Iniziamo col mostrare come ottenere le credenziali token per utilizzare le API.

# Creazione dell’account Microsoft

Per poter ottenere il token bisogna prima essere in possesso di un account Microsoft. L’account Microsoft è una combinazione di indirizzo di posta elettronica e password usata per accedere a servizi quali Outlook.com, Skype, SkyDrive, oltre che per dispositivi come i PC e tablet Windows 8, telefoni Windows Phone e console Xbox. Se si utilizza già una indirizzo di posta elettronica e una password per accedere a questi servizi, si possiede già un account Microsoft.

Nel caso in cui non si abbia un account Microsoft, è possibile crearlo al seguente indirizzo <http://login.live.com/> cliccando su *Iscriviti ora*.

# Creazione dell’account sul Marketplace di Windows Azure

Per ottenere l’accesso alle API dobbiamo accedere al Marketplace Microsoft Azure, accessibile al seguente indirizzo <https://datamarket.azure.com/>. Anche qui c’è bisogno di un account, ma questa volta non dobbiamo creare un nuovo account, ma basta collegare il nostro account Microsoft a quello del Marketplace seguendo i seguenti passi:

1. Registrare un account sul Marketplace Windows Azure (nel caso in cui si abbia già un account, si può utilizzare quello).
2. Iscriversi all’API utilizzando l’account con cui ci si è registrati.
3. Iscrivere la propria applicazione sul Marketplace;
4. Infine si ottengono le credenziali Client ID e Client Secret dell’applicazione che è stata registrata.

Vediamo le varie schermate che appariranno durante questi passi.

Quando accediamo all’indirizzo <https://datamarket.azure.com/> verrà mostrata la seguente pagina:



Figura 1 - Schermata principale Marketplace

Una volta arrivati a questa pagina bisogna accedere al nostro account cliccando su *Accedi*, posto in alto a destra. Cliccandoci verremo mandati alla schermata di login del Marketplace, ossia:



Figura 2 - Schermata di login del Marketplace

Effettuando il login, verrà mostrata la schermata di registrazione dell’account:

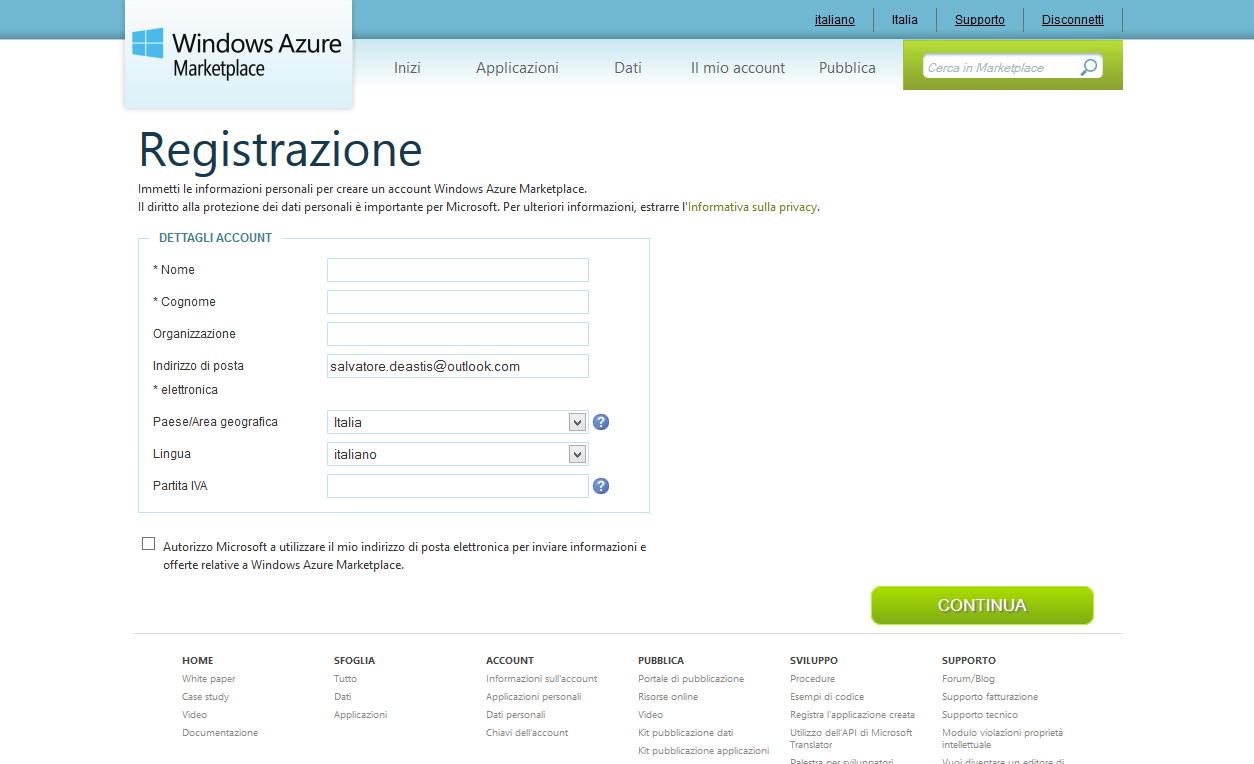


Figura 3 - Schermata di Registrazione

Inserite le credenziali, diventa cliccabile il tasto *Continua*, e accettando le condizioni per l’utilizzo del servizio, possiamo terminare la registrazione.



Figura 4 - Schermata di accettazione delle condizioni

Dopo la registrazione, il sistema ci autenticherà automaticamente al Marketplace facendo comparire il nostro nome nel menu in alto a destra.



Figura 5 - Menù utente autenticato

Adesso dobbiamo trovare le Microsoft Translator API; quindi scriviamo nella Search box ‘Translator’ e premiamo invio. Quando il sistema troverà le API le selezioniamo.

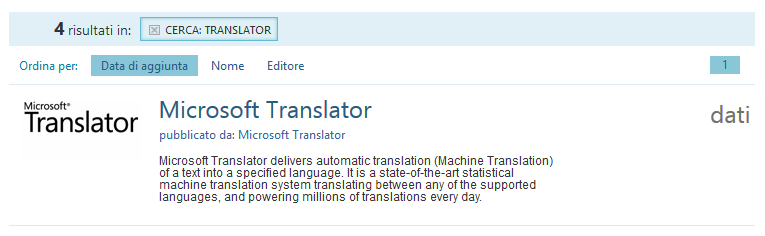


Figura 6 - Risultato della ricerca

Cliccando sul Microsoft Translator API (mostrato in figura 6), verremo portati ad una pagina contenente varie iscrizioni per utilizzare le API. Ci sono vari tipi di iscrizioni mensili; ciò che differenzia le offerte è nel numero di caratteri traducibili al mese. È disponibile anche un’iscrizione gratuita che permette di tradurre fino a 2.000.000 di caratteri/mese.

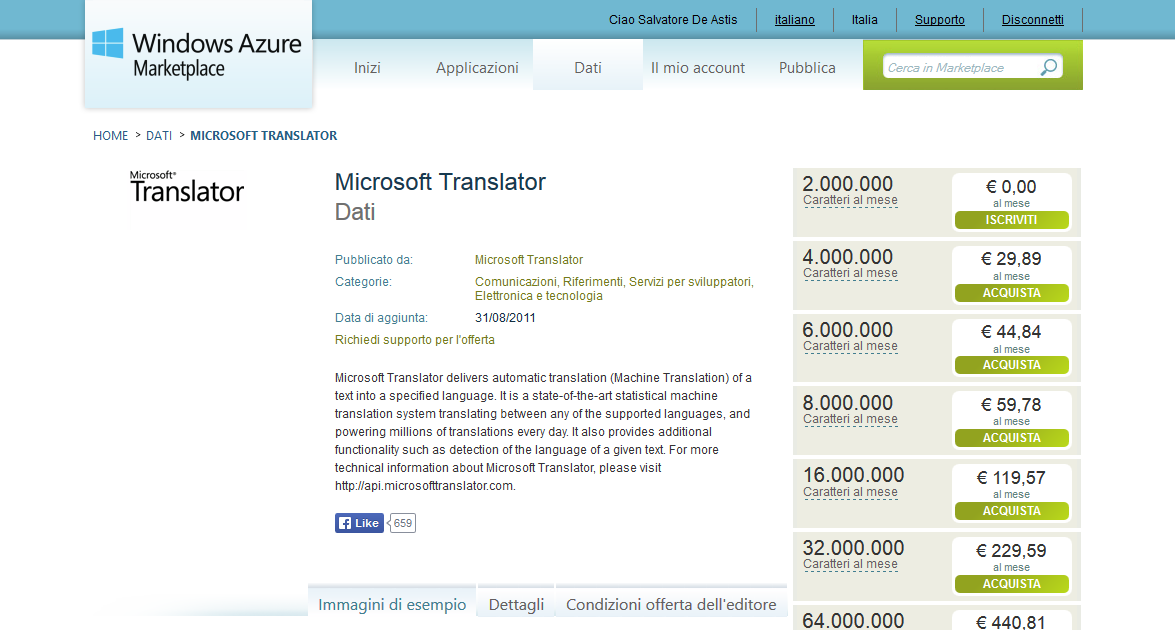


Figura 7 - Schermata di scelta del tipo di iscrizione al servizio

Cliccando su bottone ‘*Iscriviti*’, del tipo di iscrizione scelto, verremo mandati alla pagina di riepilogo dell’iscrizione scelta e la data d’inizio della decorrenza del servizio. Una volta accettate le condizioni, il tasto ‘*Iscriviti*’ diventerà verde e cliccandoci su saremo iscritti al servizio.

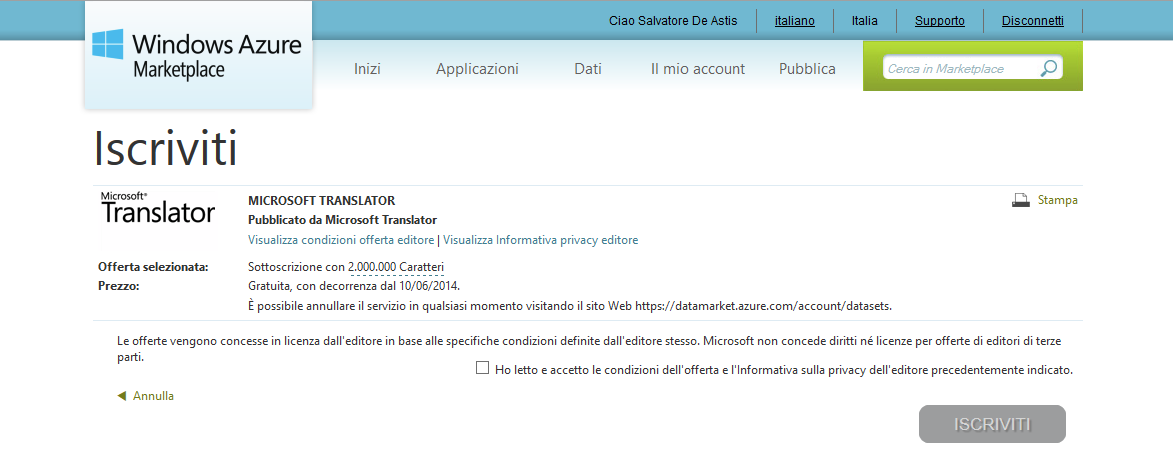


Figura 8 - Iscrizione al servizio

Una volta finita l’iscrizione, dobbiamo andare a registrare la nostra applicazione così da ottenere le nostre credenziali. Quindi andiamo alla pagina iniziale del Marketplace (quella di figura 1 accessibile al seguente link <https://datamarket.azure.com/>), accediamo al nostro account, sempre che non lo siamo già, ed andiamo alla fine della pagina così da poter vedere dei link. Sotto la voce ‘*Sviluppo*’ troveremo la voce ‘*Registra l’applicazione creata*’.

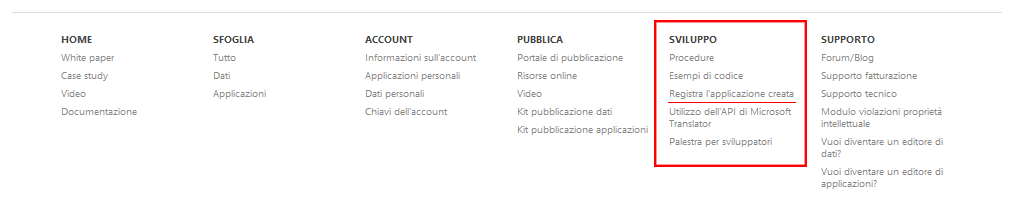


Figura 9 - Menù a fondo pagina

Verremo portati alla pagina di registrazione della nostra applicazione; questa pagina consentirà di ottenere i valori del Client ID e il Client Secret necessari per autenticare il servizio quando si genera l’applicazione. La pagina di registrazione dell’applicazione sarà simile a questa:



Figura 10 - Registrazione dell'applicazione

Vediamo cosa indicano nello specifico i campi presenti nella registrazione:

* ID Cliente: indica l’ID dell’applicazione.
* Nome: indica il nome dell’applicazione che viene visualizzato all’utente quando viene chiesta l’autorizzazione dell’utente per accedere all’account.
* Segreto cliente: è la password dell’applicazione; bisogna tenerla segreta. Questo campo viene riempito automaticamente e non deve essere cambiato.
* URI di re-indirizzamento: Indica URL da reindirizzare alla fine del flusso di consenso. Affinché un’applicazione possa accedere al Marketplace per conto di un utente, l’utente deve fornire il consenso per l’accesso dell’applicazione ai relativi account e sottoscrizioni. Se l’utente fornisce il consenso, l’applicazione ottiene un token di accesso che verrà utilizzato per eseguire l’autenticazione con i servizi di Marketplace per conto dell’utente. Microsoft Translator non utilizza il flusso di consenso, è necessario mettere un URL valido.
* Descrizione: descrizione facoltativa dell’applicazione che è visibile solo all’utente.

Una volta riempiti i vari campi, clicchiamo su ‘*Crea*’ così che la nostra applicazione venga registrata. Saremo indirizzati alla pagina Sviluppatori, dove ci saranno tutte le applicazioni che abbiamo registrato. Per ogni applicazione verranno indicati ID cliente, il nome, la descrizione e lo stato. In più saranno presenti due link:

* Modifica: che serve per ritornare alla schermata di figura 10, così da poter modificare i campi Nome, Segreto cliente, URI di re-indirizzamento e Descrizione.
* Elimina: per eliminare l’applicazione registrata.

Il tasto registra serve per poter registrare una nuova applicazione.



Figura 11 - Schermata delle applicazioni registrate

Adesso possiamo usare queste credenziali per far sì che la nostra applicazione possa utilizzare le API di Microsoft Translator.

# Implementazione dell’applicazione ‘Microsoft Traduttore’

Per semplificare la progettazione del traduttore, si è deciso di separare le varie funzionalità in moduli, così da poterli richiamare nella pagina principale dell’applicazione. In tutto sono stati creati sei moduli, ovvero:

* credential.php;
* AccessTokenAuthentication.php;
* HttpTranslator.php;
* functions\_js.js;
* css.css;
* translate.php.

Vediamo nel dettaglio ogni modulo.

# Modulo ‘credential.php’

Il modulo *credential.php* è il modulo del progetto in cui sono contenute le credenziali di accesso per l’utilizzo delle API. Ricordo che le credenziali ottenute durante la fase di registrazione sono: Client ID, Client Secret, OAuth URL, Application scope URL e l’application grant.

Il valore di ognuna di queste credenziali è stato salvato in una variabile PHP; si devono avvalorare queste variabili con i valori presenti nella schermata di registrazione dell’applicazione, vista in figura 10.

Vediamo come si presenta il modulo:

|  |
| --- |
| 1. <?php 3. $clientID = "DeAstisMicrosoftTraduttore2014"; 4. $clientSecret = "Vfx6Ihf8PoK8OEFx0ZFPgChvTR+N5pD2BLqWSe+Fp7k="; 5. $authUrl = "https://datamarket.accesscontrol.windows.net/v2/OAuth2-13/"; 6. $scopeUrl = "http://api.microsofttranslator.com"; 7. $grantType = "client\_credentials"; 9. ?> |

Modulo - credential.php

Come si vede dal codice, questo modulo è molto semplice, infatti è presente solo una dichiarazione di variabili a cui sono state associati i relativi valori. Si è deciso di mettere queste variabili in un modulo a parte così da semplificare un’eventuale modifica di questi valori, ma al contempo, permette di non inserire queste credenziali all’interno della pagina principale dell’applicazione. In particolar modo i valori delle variabili ‘*$clientID*’ e ‘*$clientSecret*’ sono molto importanti perché sono dati in modo confidenziale all’autore dell’applicazione e devono rimanere segreti rispetto all’utilizzatore dell’applicazione.

# Modulo ‘AccessTokenAuthentication.php’

L’API di Microsoft Translator per essere utilizzato, richiede l’autenticazione al Windows Azure Marketplace. In questo modulo si va a creare la classe AccessTokenAuthentication 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. Vediamo il codice di questo modulo:

|  |
| --- |
| 1. <?php 2. class AccessTokenAuthentication 3. { 4. function getTokens($grantType, $scopeUrl, $clientID, $clientSecret, $authUrl) 5. { 6. try 7. { 8. $ch = curl\_init(); 9. $paramArr = array ('grant\_type'=>$grantType, 'scope'=>$scopeUrl, 'client\_id'=>$clientID, 'client\_secret'=>$clientSecret); 11. $paramArr = http\_build\_query($paramArr); 12. curl\_setopt($ch, CURLOPT\_URL, $authUrl); 13. curl\_setopt($ch, CURLOPT\_POST, TRUE); 14. curl\_setopt($ch, CURLOPT\_POSTFIELDS, $paramArr); 15. curl\_setopt ($ch, CURLOPT\_RETURNTRANSFER, TRUE); 16. curl\_setopt($ch, CURLOPT\_SSL\_VERIFYPEER, false); 18. $strResponse = curl\_exec($ch); 20. $curlErrno = curl\_errno($ch); 21. if($curlErrno) 22. { 23. $curlError = curl\_error($ch); 24. throw new Exception($curlError); 25. } 27. curl\_close($ch); 29. $objResponse = json\_decode($strResponse); 30. if ($objResponse -> error) 31. { 32. throw new Exception($objResponse -> error\_description); 33. } 34. return $objResponse -> access\_token; 35. } catch (Exception $e)   {  echo "Exception-".$e -> getMessage();  }   1. } 2. } 3. ?> |

Modulo - AccessTokenAuthorization.php

Dopo aver definito il nome della classe, si va a dichiarare la funzione *getTokens* che prende in input il valore delle credenziali inseriti nel modulo ‘*credential.php*’.

Nella funzione viene, prima di tutto, inizializzata una sessione *cURL* utilizzando il comando *curl\_init()*. cURL è un comando che appartiene alla libreria ***libcurl***, libreria free e facile da usare da lato client, che permette il trasferimento dell’URL. Supporta famosi 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 va ad inizializzare un array contenente le credenziali di accesso passate come input alla funzione. Il contenuto dell’array verrà utilizzato dalla funzione *http\_build\_query()* che consentirà 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 va a chiudere la sessione cURL (chiamando la funzione *curl\_close()*) ed infine si va a decodificare la stringa JSON ricevuta. Viene effettuato un’ulteriore controllo, questa volta sulla stringa JSON, dove in caso di errore viene sollevata un’eccezione altrimenti verrà restituito il token di accesso.

# Modulo ‘HttpTranslator.php’

Nel modulo è presente l’altra classe helper che consente di gestire la comunicazione con le API al nostro posto. Viene utilizzata la funzionalità cURL di PHP così può comunicare in maniera sincrona con un servizio web di tipo HTTP.

Il codice di questo modulo è il seguente:

|  |
| --- |
| 1. <?php 3. Class HTTPTranslator 4. { 5. function curlRequest($url, $authHeader) 6. { 7. $ch = curl\_init(); 8. curl\_setopt ($ch, CURLOPT\_URL, $url); 9. curl\_setopt ($ch, CURLOPT\_HTTPHEADER, array($authHeader,"Content-Type: text/xml")); 10. curl\_setopt ($ch, CURLOPT\_RETURNTRANSFER, TRUE); 11. curl\_setopt ($ch, CURLOPT\_SSL\_VERIFYPEER, False); 12. $curlResponse = curl\_exec($ch); 14. $curlErrno = curl\_errno($ch); 15. if ($curlErrno) 16. { 17. $curlError = curl\_error($ch); 18. throw new Exception($curlError); 19. } 21. curl\_close($ch); 22. return $curlResponse; 23. } 24. } 25. ?> |

Modulo - HttpTranslator.php

In generale la funzione prende come parametri di input il request URL, il request header e i dati mandati attraverso il metodo POST dal form e restituisce il cURL di risposta o un errore.

Una volta implementata anche questa funzione abbiamo tutte le funzioni necessarie per accedere al servizio di traduzione.

# Modulo ‘translate.php’

Possiamo considerare questo modulo come la pagina principale della nostra applicazione, in quanto contiene non solo gli elementi che consentono di inserire il testo da tradurre e di scegliere la lingua di origine del testo e la lingua in cui vogliamo che il nostro testo sia tradotto, ma contiene anche il comportamento dei due bottoni principali: ‘*Traduci*’ e ‘*Ascolta*’. Quest’ultimo bottone è presente nella pagina con il simbolo di un megafono vicino al bottone ‘*Traduci*’.

Nella modulo sono presenti due sezioni principali:

1. la prima è la sezione **PHP**, indicata con i tag *<?php ?>*, in cui è presente tutto il codice necessario per effettuare la traduzione e la generazione dello stream audio per il text-to-speech.
2. la seconda sezione comprende tutto il codice **HTML**, indicato con i tag *<html> </html>*, necessario per creare gli oggetti presenti nella pagina, come le textarea, select box e i vari bottoni.

Vediamo nel dettaglio le due sezioni.

Il codice della parte PHP del modulo è il seguente:

|  |
| --- |
| 1. <?php 2. session\_start(); 3. if (glob('./speech\_file/\*.mp3')) 4. { 5. $file = glob('speech\_file/'.'\*.mp3',GLOB\_MARK); 6. foreach ($file as $file) 7. { 8. if (is\_dir($file)) 9. { 10. self::deleteDir($file); 11. } 12. else 13. { 14. unlink($file); 15. } 16. } 17. } 18. include 'HttpTranslator.php'; 19. include 'AccessTokenAuthentication.php'; 20. require 'credential.php'; 21. if (isset($\_POST['traduci\_sub'])) 22. { 23. if (isset($\_POST['source\_lang'])) 24. { 25. $\_SESSION['source\_lang'] = $\_POST['source\_lang']; 26. } 27. if (isset($\_POST['dest\_lang'])) 28. { 29. $\_SESSION['dest\_lang'] = $\_POST['dest\_lang']; 30. } 32. try 33. { 34. $authObj = new AccessTokenAuthentication(); 35. $accessToken = $authObj -> getTokens($grantType, $scopeUrl, $clientID, $clientSecret, $authUrl);     1. $authHeader = "Authorization: Bearer ". $accessToken;     2. $fromLanguage = $\_POST["source\_lang"];     3. $toLanguage = $\_POST["dest\_lang"];     4. $inputStr = $\_POST["source\_text"];     5. $contentType = 'text/plain';     6. $category = 'general';     7. $paramst = "text=".urlencode($inputStr)."&to=".$toLanguage."&from=".$fromLanguage;     8. $translateUrl = "http://api.microsofttranslator.com/v2/Http.svc/Translate?$paramst";     9. $translatorObj = new HTTPTranslator();     10. $curlResponse = $translatorObj -> curlRequest($translateUrl, $authHeader);     11. $xmlObj = simplexml\_load\_string($curlResponse);     12. foreach((array)$xmlObj[0] as $val)     13. {         1. $translatedStr = $val;     14. }     15. $translatedText = urlencode($translatedStr);     16. $out = 'audio/mp3';     17. $params = "text=$translatedText&language=$toLanguage&format=$out";     18. $url = "http://api.microsofttranslator.com/V2/Http.svc/Speak?$params";     19. $translatorObj = new HTTPTranslator();     20. $strResponse = $translatorObj -> curlRequest($url, $authHeader);     21. if (!is\_dir('speech\_file'))     22. {         1. mkdir('speech\_file');     23. }     24. $var = uniqid('SPC\_').".mp3";     25. file\_put\_contents('./speech\_file/'.$var, $strResponse); 36. } catch (Exception $e) 37. {     1. echo "Exception: " . $e->getMessage() . PHP\_EOL; 38. } 39. } 40. ?> |

Modulo - translate.php (parte php)

Per poter utilizzare le API per effettuare la traduzione e il text-to-speech, la richiesta avviene tramite il passaggio del testo e della lingua di origine e di destinazione. Inizialmente viene effettuato 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; questo viene fatto per permettere all’applicazione di mantenere la scelta delle lingue effettuata dall’utente anche dopo il refresh della pagina.

Arrivati a questo punto inizia la procedura per poter tradurre il testo: il tutto viene eseguito in un *try-catch* così che nel caso in cui ci sia un errore, esso venga riportato a schermo. Si inizia con l’ottenere il permesso per utilizzare le API ottenendo un token. Successivamente attraverso il passaggio dei valori del testo e delle lingue si vanno a settare i parametri che saranno aggiunti all’URL per poter eseguire la traduzione. 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*.

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*’ e 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’, ma API mette a disposizione anche il formato ‘.wav’; nel caso in cui il formato non venisse specificato, l’API utilizza il formato ‘.wav’. Un altro parametro che si può settare riguarda la qualità dell’audio, non utilizzato nella implementazione di questo traduttore.

Una volta che si è 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.

Per quanto riguarda la sezione **HTML** non ha bisogno di particolari spiegazioni, in quanto contiene il codice necessario per la creazione degli oggetti del form come textarea, select box e i vari bottoni. L’unica parte del codice che potrebbe essere necessario spiegare è quello riguardante le select box.

Come già specificato nelle select box sono presenti le lingue disponibili per la traduzione (quelle presenti nell’applicazione non sono tutte quelle disponibili, l’elenco con tutte le lingue messe a disposizione dell’API è disponibile al seguente indirizzo <http://msdn.microsoft.com/en-us/library/hh456380.aspx>). Per non creare delle select box con un numero elevato di tag ‘*<option>*’ e per non duplicare il codice HTML per riempire le select box, si è deciso di dichiarare un array contenente le lingue.

L’array è stato settato in modo tale che l’indice di ogni elemento corrispondesse al codice della lingua utilizzato dall’API. Così facendo il valore che viene passato allo script PHP, spiegato in precedenza, può essere direttamente inserito nell’URL senza che sia richiesto il ritrovamento del codice della lingua attraverso una richiesta all’API.

Dichiarato l’array, nelle select è stato necessario inserire del codice PHP sia per poter accedere all’array e visualizzarne i valori, sia per far in modo che se l’utente ha già fatto una traduzione, vengano ricordate le scelte precedenti. Inoltre è possibile anche scegliere le altre lingue per traduzioni successive.

Questo comportamento è stato implementato tramite condizioni in PHP e attraverso l’utilizzo delle variabili di sessione: all’inizio viene controllato se la variabile di sessione sia vuota o contenga un valore. Nel caso in cui fosse vuota allora si devono mostrare all’utente tutte le lingue disponibili nell’array delle lingue, altrimenti viene fatta una ricerca nel vettore finché non venga trovata la lingua che abbia il codice uguale a quello presente nella variabile di sessione. Siccome l’utente potrebbe decidere di voler cambiare lingua, è stato aggiunto il codice in modo da mostrare anche le altre lingue una volta che apre il menu a tendina della select box.

|  |
| --- |
| 1. <!DOCTYPE html> 2. <html lang = "it"> 3. <head> 4. <meta charset="utf-8" /> 5. <title>Microsoft Translator</title> 6. <link rel = "stylesheet" href = "css.css" /> 7. <script type = "text/javascript" src = "functions\_js.js"></script> 8. </head> 9. <body> 11. <?php 13. $languages = array("a"=>"Seleziona lingua", "it"=>"Italiano", "en"=>"Inglese", "ja"=>"Giapponese", "pt"=>"Portoghese", "es"=>"Spagnolo", "tr"=>"Turco","zh-CHS"=>"Cinese Semplificato", "zh-CHT"=>"Cinese Tradizionale", "de"=>"Tedesco", "fr"=>"Francese",); 14. Ksort($languages); 15. ?> 17. <div class = "body"> 18. <div class = "intestazione"> 19. <img class = "logo" src = "image/logo.png" width = "300" height = "47" alt = "logo"/> 20. </div> 22. <p class = "titolo">Traduttore</p> 23. <form id ="trad" name = 'trad' action = "translate.php" method = "POST"> 24. <select class = "source\_lang" name = "source\_lang"> 25. <?php if ($\_SESSION['source\_lang'] == NULL) 26. { 27. foreach($languages as $code => $lang) 28. {?> 29. <option value = <?php echo $code; ?>"> <?php echo $lang; ?> </option> 30. <?php } 31. } 32. else 33. { 34. foreach($languages as $code => $lang) 35. { if ($code == $\_SESSION['source\_lang']) 36. {?> 37. <option value = "<?php echo $code; ?>"> <?php echo $lang; ?> </option> 38. <?php } 39. } 40. foreach($languages as $code => $lang) 41. {?> 42. <option value = "<?php echo $code; ?>"> <?php echo $lang; ?> </option> 43. <?php } 44. } ?> 45. </select> 47. <textarea id = "source\_text" class = "source\_text" name = "source\_text" onclick = "javascript:select(source\_text)" onkeypress = "javascript:send('traduci\_sub', event);"> <?php if (isset($inputStr)== true){echo $inputStr;}else{echo '';} ?></textarea> 48. <select id = "dest\_lang" class = "dest\_lang" name = "dest\_lang"> 50. <?php if ($\_SESSION['source\_lang'] == NULL) 51. { 52. foreach($languages as $code => $lang) 53. { ?> 55. <option value = "<?php echo $code; ?>"> <?php echo $lang; ?> </option> 56. <?php } 57. } 58. else 59. { 60. foreach($languages as $code => $lang) 61. { if ($code == $\_SESSION['dest\_lang']) 62. { ?> 63. <option value = "<?php echo $code; ?>"> <?php echo $lang; ?> </option> 64. <?php } 65. } 66. foreach($languages as $code => $lang) 67. { ?> 68. <option value = "<?php echo $code; ?>"> <?php echo $lang; ?> </option> 69. <?php } 70. }?> 72. </select> 73. <input id = "traduci\_sub" class = "traduci\_sub" name ="traduci\_sub" type = "submit" value = "Traduci"> 74. </form> 75. <textarea id = "transl\_text" class = "transl\_text" name = "transl\_text" onclick = "javascript:select(transl\_text)"><?php if (isset($inputStr)== true){echo $translatedStr;}else{echo '';} ?></textarea> 76. <button class = "reset" onclick = "javascript:trad.source\_text.value = ''; javascript:transl\_text.value = ''">Reset</button> 78. <?php foreach(glob('./speech\_file/\*.\*') as $filename) 79. { 80. $speech = $filename; 81. } 82. ?> 83. <audio id = "t2s" controls = "controls" > 84. <source src = "<?php echo $speech; ?>" type = "audio/mpeg" /> 85. </audio> 86. <div class = "pagef"> 87. <img class = "credit" src = "image/credit.png" width = "200" height = "25" alt = "credit"/> 88. </div> 89. </div> 90. </body> 91. </html> |

Modulo - translate.php (parte HTML)

Una volta premuto il tasto ‘*Traduci*’ sotto la textarea dove compare il testo tradotto, comparirà una barra audio, ottenuta utilizzando il tag ‘*<audio>*’ messo a disposizione dall’ HTML 5, in modo tale che l’utente possa ascoltare il testo tradotto.

Il tasto ‘*Reset*’ ha il compito di pulire le text area da ciò che contiene; questo comportamento è stato ottenuto utilizzando una funzione Javascript che si attiva ogni volta che il tasto viene premuto. La funzione è molto semplice in quanto imposta il valore della textarea come vuoto; il valore vuoto nella funzione viene impostato tramite **‘ ’** (apici senza nessun contenuto all’interno di esse).

# Modulo ’functions\_js.js’

Il modulo functions\_js.js è un modulo contenente due funzioni Javascript che vengono utilizzate nel modulo ‘*translate.php*’, in cui viene incluso. Le funzioni che sono presenti nel modulo sono:

* select: questa funzione permette di selezionare il testo presente nella textarea quando si clicca su quest’ultima.
* send: questa funzione permette di inviare il testo da tradurre, quindi di spingere il pulsante ‘*Traduci*’ premendo il tasto ‘*Invio*’ (‘*Enter*’ nelle tastiere inglesi/americane) della tastiera.

Il modulo si presenta in questa maniera:

|  |
| --- |
| 1. function select(text) 2. { 3. var t = document.getElementById(text).innerHTML; 4. alert(t); 5. } 7. function send(t, e) 8. { 9. if ((e.which && e.which == 13) || (ev.keyCode && e.keyCode == 13)) 10. { 11. document.getElementById(t).click(); 12. return false; 13. } 14. else 15. { 16. return true; 17. } 18. } |

Modulo – functions\_js.js

# Modulo ‘css.css’

Questo è il foglio di stile in cui sono presenti tutte le direttive per la formattazione dei componenti dell’applicazione. Questo file viene incluso nel modulo ‘*translate.php*’.

|  |
| --- |
| 1. *@font-face* 2. { 3. font-family: 'Segoe UI Light'; 4. src: url(*'font/segoe\_ui\_light.eot'*); 5. src: url(*'font/segoe\_ui\_light.eot?#iefix'*) format('embedded-opentype'), 6. url(*'font/segoe\_ui\_light.svg#Segoe UI Light'*) format('svg'), 7. url(*'font/segoe\_ui\_light.woff'*) format('woff'), 8. url(*'font/segoe\_ui\_light.ttf'*) format('truetype'); 9. font-weight: normal; 10. font-style: normal; 11. } 12. div.intestazione 13. { 14. position: absolute; 15. width: 100%; 16. height: 10%; 17. top: 0%; 18. left: 0%; 19. background-color: #eeecec; 20. } 21. img.logo 22. { 23. position: absolute; 24. top: 6px; 25. } 27. p.titolo 28. { 29. position: absolute; 30. top: 10%; 31. font-family: Segoe UI; 32. font-size: 25px; 33. font-style: normal; 34. color: #f00; 35. } 36. select.source\_lang 37. { 38. position:absolute; 39. top: 23%; 40. width: 146px; 41. } 42. select.dest\_lang 43. { 44. position:absolute; 45. top: 23%; 46. left: 50%; 47. width: 146px; 48. } 49. select 50. { 51. font-size: 14px; 52. color: #333; 53. background: #CCC; 54. font-family: Segoe UI; 55. border: solid 2px; 56. border-color:#999; 57. } 58. button.reset 59. { 60. position: absolute; 61. top: 65%; 62. left: 8px; 63. font-size: 14px; 64. color: #333; 65. background: #CCC; 66. font-family: Segoe UI; 67. border: solid 2px; 68. border-color:#999; 69. } 70. input.traduci\_sub 71. { 72. position: absolute; 73. top: 23%; 74. left: 65%; 75. width: 100px; 76. font-size: 14px; 77. color: #fff; 78. background: #0094ff; 79. font-family: Segoe UI; 80. border: solid 2px; 81. border-color:#0082ff; 82. } 84. audio 85. { 86. position: absolute; 87. top: 57%; 88. left: 50.5%; 89. } 90. textarea.source\_text 91. { 92. position:absolute; 93. top: 30%; 94. font-family: 'Segoe UI'; 95. } 97. textarea.transl\_text 98. { 99. position:absolute; 100. top: 30%; 101. left: 50%; 102. font-family: 'Segoe UI'; 103. } 104. textarea 105. { 106. font-size: 14px; 107. width: 440px; 108. height: 200px; 109. border:solid 2px; 110. border-color: #999; 111. resize: none; 112. } 113. div.pagef 114. { 115. position: absolute; 116. width: 100%; 117. height: 10%; 118. bottom: 0; 119. left: 0; 120. background-color: #eeecec; 121. } 122. img.credit 123. { 124. position: absolute; 125. bottom: 16px; 126. left: 80%; 127. } |

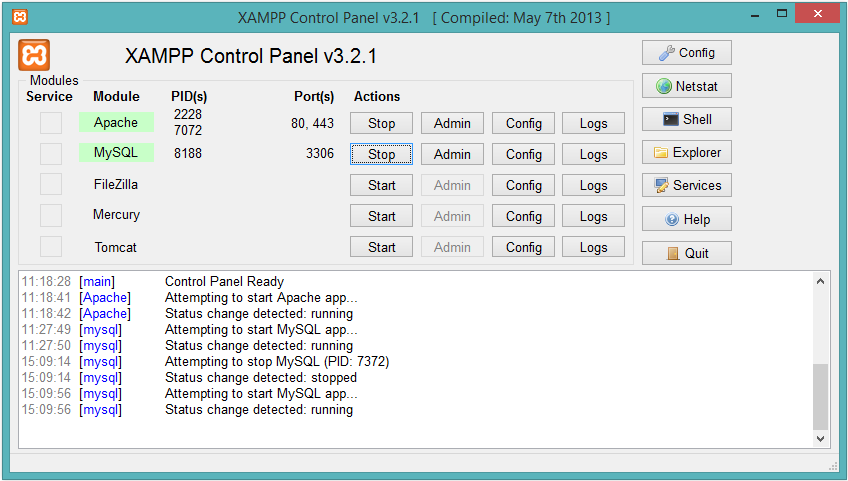
Modulo – css.css

# Installazione dell’applicazione in XAMPP su piattaforma Windows

È possibile utilizzare l’applicazione utilizzando il server web XAMPP. XAMPP è una piattaforma software gratuita costituita da Apache http Server, MySQL e da tutti gli altri strumenti necessari per utilizzare linguaggi di programmazione come PHP e Perl.

Per poter utilizzare l’applicazione, basta installare XAMPP, disponibile al seguente indirizzo <https://www.apachefriends.org/index.html>e, una volta installata, basta inserire la cartella dell’applicazione nella cartella ‘*htdocs*’ di XAMPP raggiungibile al seguente percorso ‘*C:/xampp/*’. Si noti che il percorso potrebbe cambiare se durante la fase di installazione del web server si è scelto un percorso differente.

Per poter avviare l’applicazione dobbiamo attivare il servizio Apache e MySQL, andando ad utilizzare l’interfaccia di XAMPP. Il servizio sarà attivo se la scritta del servizio sarà evidenziata di verde come mostrato in figura.



Attivazione servizio Apache in XAMPP

Una volta attivati i servizi, dobbiamo verificare XAMPP sia realmente attivo; per far questo dobbiamo aprire il browser e digitare l’indirizzo ‘*http://localhost/*’. Se il tutto funziona dovrebbe apparire la schermata di benvenuto come la seguente (si noti che la schermata potrebbe cambiare in base alla versione).



Schermata iniziale di XAMPP

Ora per accedere all’applicazione basta inserire il percorso nella barra dell’indirizzo; quindi basta aggiungere all’URL il nome della cartella contenente il nome del progetto e successivamente il nome della pagina principale del progetto, che è ‘*translate.php*’. Quindi l’URL diventa: ‘*http://localhost/speech-translation-tools/translate.php*’. Vediamo come appare la schermata iniziale dell’applicazione:



Pagina iniziale del Traduttore

Adesso l’utente può inserire il testo che vuole tradurre nella textarea presente a sinistra dello schermo, scegliere la lingua di origine e quella di festinazione e premere sul pulsante ‘*Traduci*’. Alla fine verrà mostrato nella textarea a destra il testo tradotto e comparirà il player per sentire il text-to-speech del testo tradotto.



Esempio di traduzione

# Installazione dell’applicazione in XAMPP su piattaforma Linux

Per quanto riguarda l’installazione dell’applicazione su piattaforma linux (si è testato l’applicazione su Ubuntu 14.04 a 64 bit) il procedimento non cambia di molto. Innanzitutto bisogna scaricare XAMPP; una volta scaricato dobbiamo cambiare i permessi al file utilizzando il seguente comando:

sudo chmod 755 xampp-linux-\*-installer.run

e avviare l’installazione utilizzando questo comando:

sudo ./xampp-linux-\*-installer.run

Una volta finita la procedura, XAMPP sarà disponibile nella seguente directory ‘*/opt/lampp*’. Dopo bisogna attivare i servizi Apache e MySQL; questo viene fatto utilizzando il comando:

sudo /opt/lampp/lampp start

Quando eseguiremo il comando, nel terminale comparirà un messaggio simile al seguente:

Starting XAMPP for Linux 1.8.1...

XAMPP: Starting Apache with SSL (and PHP5)...

XAMPP: Starting MySQL...

XAMPP: Starting ProFTPD...

XAMPP for Linux started.

Adesso per verificare che I servizi siano effettivamente avviati, basta aprire il browser e digitare nella barra degli indirizzi ‘http://*localhost/*’. Se tutto è andato a buon fine apparirà la schermata principale di XAMPP. Adesso basterà inserire la cartella dell’applicazione nel percorso ‘*/opt/lampp*’ e digitare nella barra degli indirizzi ‘*http://localhost/speech-translation-tools/transalate.php*’.

Se si volesse terminare i servizi basterà digitare nel terminale il comando:

sudo /opt/lampp/lampp stop

Buona traduzione!