Creare applicazioni multilingue

Indice generale

Creare applicazioni multilingue

Utilizza il sistema delle traduzioni di Instant Developer Cloud per sviluppare applicazioni multilingue.

Introduzione

L'adattamento di un'applicazione ad un contesto multilingue è un compito complesso. I principali punti di intervento, in funzione della lingua della sessione, sono i seguenti:

- Modifica delle proprietà degli elementi visuali e delle liste valori.
- Modifica delle stringhe contenute nel codice.
- Modifica dei formati di input e output dei numeri e delle date.
- Modifica delle descrizioni di entità estratte dal database.

Ovviamente non tutte le proprietà, le liste valori, le stringhe, i formati e le descrizioni devono risultare tradotte, solo quelle che si riferiscono ad un contesto multilingue.

Per ottenere questo risultato occorre un sistema in grado di localizzare tutte le istanze di stringhe da tradurre, permettere ai traduttori di effettuare la traduzione conoscendo il contesto, infine un sistema che adatti l'output dell'applicazione verso il browser selezionando i dati di traduzione relativi alla lingua di ogni diversa sessione di lavoro.

L'approccio tradizionale

L'approccio tradizionale, possibile anche con Instant Developer Cloud, è piuttosto semplice: utilizzare una mappa, cioè un insieme di elementi chiave-valore, per valorizzare ogni elemento visuale che contiene informazioni dipendenti dalla lingua. È quindi sufficiente un oggetto javascript definito a livello di applicazione che rappresenta le traduzioni di tutti gli oggetti traducibili, in tutte le lingue disponibili. Poi, tramite codice, si carica la traduzione corretta.

Immaginiamo di avere quindi un oggetto così costituito:

```
App.traduzioni = {
  ITEM_LABEL: {
    "it": "Articolo",
    "en": "Item",
    "ru": "пункт",
    …
  },
  …
};
```

In tutti i punti in cui ci si riferisce all'etichetta di un articolo è sufficiente scrivere la seguente riga di codice nell'evento *onLoad* oppure *onRowComposition*.

```
$itemLabel.innerText = App.traduzioni.ITEM_LABEL[app.langCode];
```

Dove *app.langCode* è la proprietà della sessione che rappresenta il codice della lingua (“it”, “en”, eccetera).

Si potrebbe considerare anche un approccio più generale, creando un metodo che scorre la struttura degli elementi visuali di una videata e applica tutte le traduzioni in base al nome dell’elemento, ma come vedremo in seguito, questa non è un'opzione percorribile.

Problemi dell’approccio tradizionale

I problemi che l’approccio tradizionale pone sono molti. Vediamo i principali:

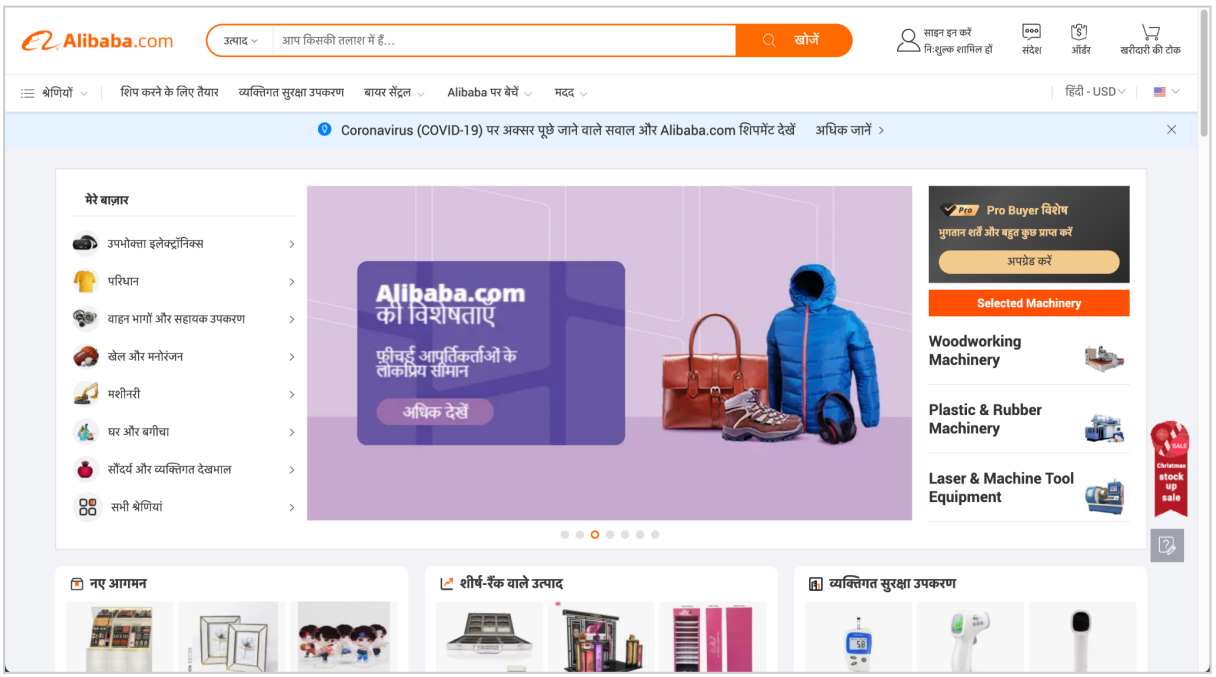
1) Durante lo sviluppo è necessario decidere da subito quali sono le informazioni che devono essere localizzate. Questa informazione, invece, non è così scontata ed è facile dimenticarsene, considerato che anche in un'applicazione costituita da dieci videate si può facilmente arrivare ad avere centinaia di stringhe localizzabili.
2) Non è facile decidere se la medesima informazione deve avere lo stesso codice di traduzione o meno: dovendo rendere localizzabile la label *Articolo* in diverse videate, è vero che in tutte queste si deve usare sempre la stessa costante ITEM_LABEL? Oppure devono essere tutte diverse? Questa decisione spetterebbe ad un esperto linguista, non ad un programmatore che sta scrivendo il codice di traduzione.
3) Se si hanno migliaia di proprietà traducibili, la modifica dell’oggetto di traduzione e la ricerca di proprietà già esistenti diventa impossibile o comunque molto costosa.
4) Il caso precedente fa diventare ancora più inaffidabile l’utilizzo di un metodo di traduzione generico in base ai nomi degli elementi. Essi vengono scelti con altri criteri dai programmatori, basta che siano univoci, non certo con criteri linguistici.
5) L’estrazione, la programmazione e l’implementazione dell’oggetto contenente le traduzioni è quindi un compito complesso che richiede tempo e quindi denaro.
6) Se poi l’applicazione deve diventare localizzabile dopo che essa è già stata sviluppata in modo non localizzabile, ad esempio perché cambiano le specifiche in una versione successiva, si tratta di una vera e propria riprogrammazione di almeno tutto il front-end.
7) Anche il compito dei traduttori non è facile: ricevendo un oggetto JSON come quello precedente, non hanno minimamente idea del contesto in cui le informazioni devono apparire. Quindi la precisione della traduzione sarà minima, sotto allo stato dell’arte di una traduzione professionale, tanto vale usare Google Translate.
8) Quando una stringa deve essere modificata nella lingua madre, ad esempio perché cambiano le funzioni dell’interfaccia utente, non è facile tenere traccia delle differenze fra una versione e la successiva. I traduttori dovrebbero utilizzare un software CAT (Computer-Assisted Translation) in grado di tracciare le differenze, ma questo non è sempre vero. Inoltre, l’affidabilità del tracciamento delle differenze dipende dalla struttura dei file che vengono dati in input a tali strumenti: se i file non sono strutturati in modo compatibile con i software CAT, il tracciamento in molti casi non è possibile. Comunque non è facile essere certi di non pubblicare un'applicazione parzialmente tradotta o con traduzioni non aggiornate, magari solo in alcune lingue.
9) Anche la lingua madre di un’applicazione dovrebbe essere considerata come una traduzione, infatti se il contenuto testuale dell’interfaccia viene deciso dagli

sviluppatori, occorre anche qui un esperto linguista per rendere più comprensibile l'applicazione all'utente finale.

L'approccio Instant Developer Cloud

Solitamente il problema delle traduzioni viene considerato come "poco importante" dai reparti tecnici che si occupano di sviluppo. Dal punto di vista dell'utente finale, invece, avere una applicazione non tradotta o non comprensibile è un difetto di pari gravità ad un comportamento errato, forse in alcuni casi ancora peggiore.

Per convincersene è sufficiente provare ad installare un'applicazione non localizzata in una lingua non familiare, come ad esempio russo o cinese. Non sarà possibile nemmeno iniziare ad usarla.



Anche un sito molto famoso può essere inutilizzabile se non correttamente localizzato

Visti i problemi che l'approccio tradizionale pone sia a livello di tempi e costi di sviluppo, sia, soprattutto, di precisione del risultato e vista l'importanza che l'utente finale pone su questo argomento, Instant Developer Cloud include un framework specifico e una serie di strumenti IDE ad esso dedicati.

L'obiettivo di questi strumenti è di ottimizzare il processo di localizzazione degli elementi visuali, delle liste valori e delle stringhe inserite a livello di codice, sia nell'applicazione principale che in eventuali componenti importati, risolvendo tutti i problemi indicati sopra.

I paragrafi seguenti illustrano il processo di preparazione e di traduzione. Al termine di questo libro verrà indicata una strategia per la traduzione delle descrizioni provenienti dal database.

Processo di traduzione

Durante lo sviluppo

Il sistema di traduzioni di Instant Developer Cloud è pensato per essere completamente non invasivo in relazione al processo di sviluppo.

Gli sviluppatori non devono quindi avere alcuna particolare accortezza durante lo sviluppo dell'applicazione. Non devono nemmeno sapere a priori che l'applicazione dovrà prima o poi essere localizzata.

L'unica indicazione da applicare, utile anche nel caso in cui l'applicazione non debba essere tradotta, è quella di utilizzare una particolare funzione per la composizione dei messaggi da mostrare all'utente. Vediamo un esempio. Invece che:

```
let msg = "L'utente " + nome + " non esiste";
```

È opportuno utilizzare la seguente notazione:

```
let msg = t("L'utente @nome non esiste", { nome : nome });
```

La funzione *t* permette di comporre un messaggio dato un template e un oggetto che specifica i dati da sostituire. Nel template i parametri sono indicati da @ seguita dal nome della proprietà dell'oggetto passato come dati.

In questo modo si ha il vantaggio di scrivere un codice più leggibile durante lo sviluppo. Inoltre, il messaggio è già pronto per la localizzazione. Nella fase di localizzazione infatti, è necessario fornire il massimo contesto possibile per avere una traduzione accurata. Localizzare il messaggio: "*L'utente @nome non esiste"* ha un certo contesto; tradurre separatamente le due stringhe "*L'utente"* e "*non esiste"* non fornisce lo stesso contenuto semantico ed il risultato sarà qualitativamente inferiore o addirittura errato.

Instant Developer Cloud non consente l'uso delle stringhe template JavaScript ai fini della traduzione. L'uso della funzione di composizione *t* invece permette di attivare il framework di localizzazione per il messaggio ad essa applicato.

Preparazione alla localizzazione

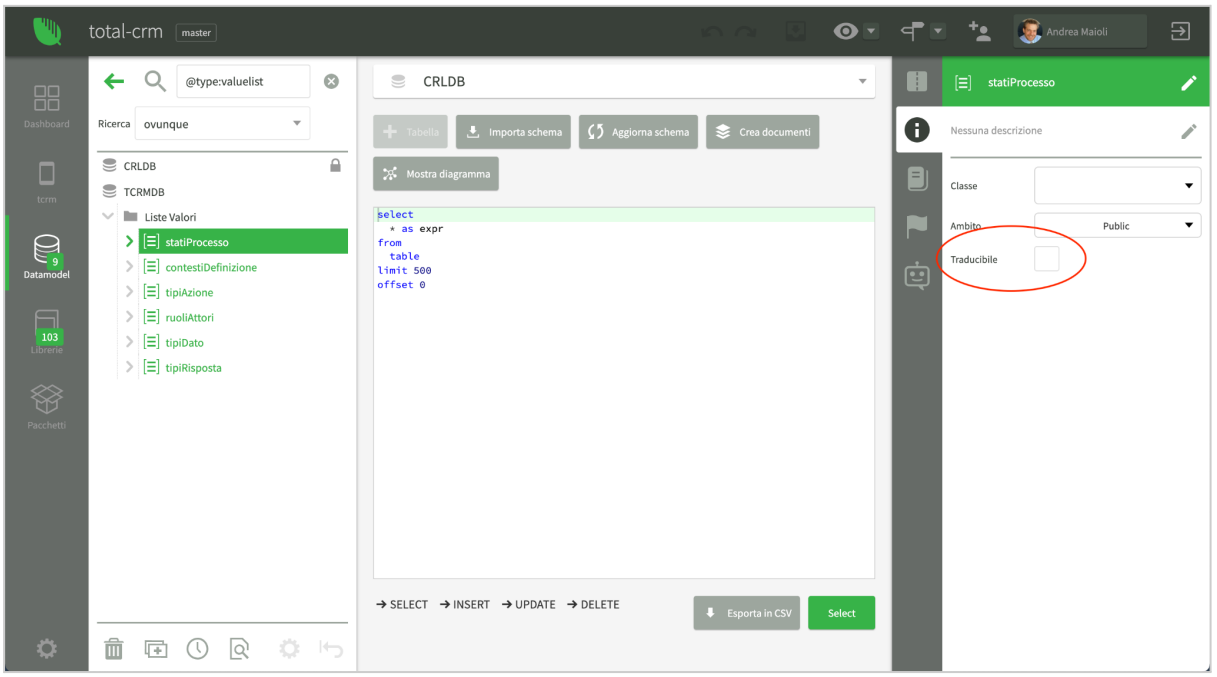
Vediamo ora quali operazioni devono essere fatte prima della fase di traduzione vera e propria. Queste operazioni devono essere effettuate tramite l'IDE di Instant Developer Cloud da parte di persone con conoscenza della struttura dell'applicazione. Non è richiesta invece esperienza di programmazione.

Selezione delle liste valori da localizzare

Le liste valori possono essere usate per mostrare la descrizione associata ad un dato dell'applicazione nell'interfaccia utente. In questi casi, la lista valori dovrà essere localizzata.

Il primo passaggio di preparazione, consiste quindi nella ricerca delle liste valori da localizzare.

Tramite la funzione di ricerca dell'IDE è possibile trovare tutte le liste valori presenti nel progetto, come si vede nell'immagine seguente.



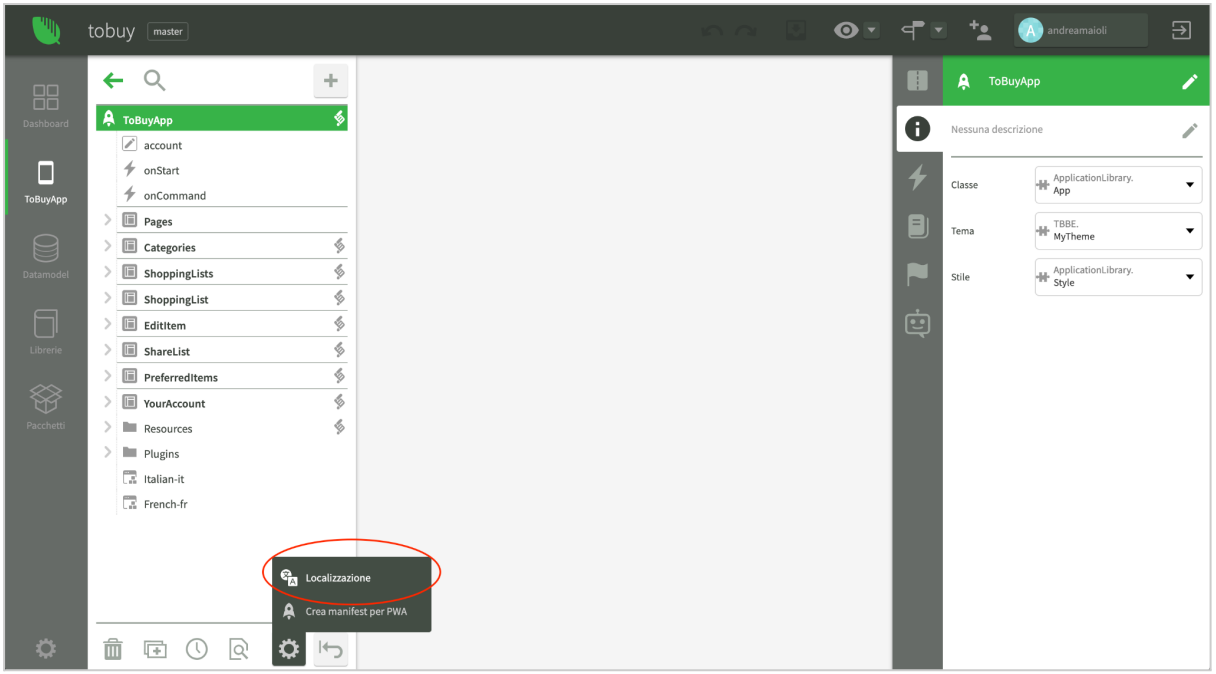
La ricerca avviene scrivendo *@type:valuelist* nel campo di ricerca ed impostando il contesto a *ovunque*. La selezione delle liste valori da tradurre deve avvenire solo negli oggetti del progetto, non in quelli dei package importati.

Selezionando una lista valori, è possibile renderla localizzabile impostando il flag *Traducibile* nella barra delle proprietà. In questo modo il contenuto della lista verrà incluso nei dati del sistema di traduzione.

Selezione delle stringhe da localizzare

La seconda operazione consiste nel selezionare le stringhe che dovranno essere localizzate rispetto a tutte quelle presenti nel sistema di traduzione. A tal fine si consiglia di operare come segue:

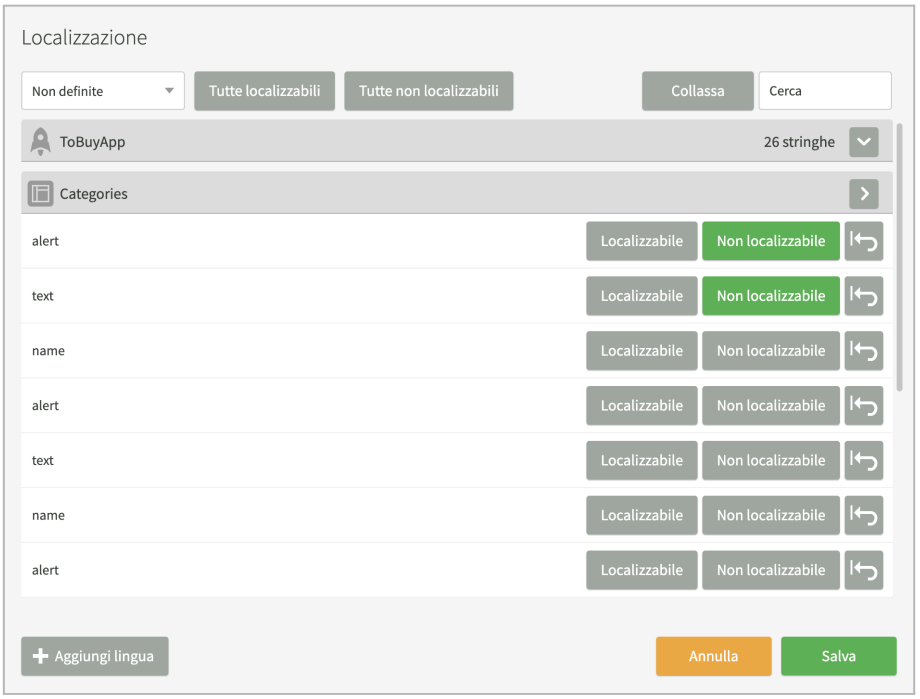
1. Attivare il sistema di traduzione, tramite il menu personalizzato *Localizzazione* quando nell'albero viene selezionata l'applicazione da localizzare.



2. Aggiungere la prima lingua in cui si desidera localizzare l'applicazione, tramite il pulsante *Aggiungi lingua* in basso nella pagina di localizzazione.

3. Dopo aver aggiunto una lingua, apparirà la lista delle stringhe non ancora definite, cioè per le quali non è ancora stato deciso se devono essere localizzate o meno. Il sistema di localizzazione estrae la lista delle stringhe dalle proprietà localizzabili degli elementi visuali, come ad esempio *innerText*, dalle stringhe nel codice gestite con la funzione *t* e dalle liste valori marcate come traducibili.

4. In questo momento è importante decidere se una stringa deve essere localizzata o meno. Per ogni stringa della lista di quelle non definite, utilizzare i pulsanti sulla riga per effettuare questa decisione. Il pulsante con l'icona freccia permette di mostrare la posizione della stringa nel progetto, quindi la videata o il metodo in cui essa appare. In questo stato è possibile cliccare in ogni punto dello schermo per ritornare alla pagina di localizzazione.

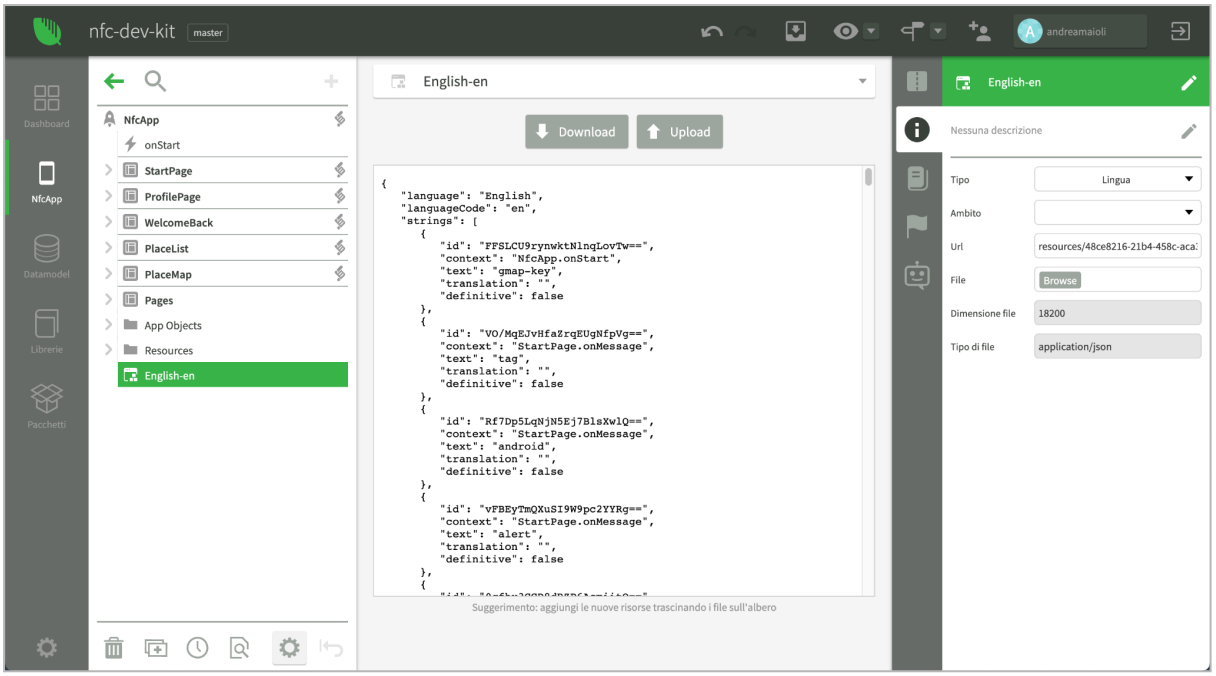


I pulsanti *Tutte localizzabili* e *Tutte non localizzabili* permettono di modificare lo stato di tutte le righe presenti a video. Essi agiscono sulla selezione presente a video, in considerazione dell'eventuale ricerca effettuata tramite il campo *Cerca* in alto a destra.

Attenzione: la selezione delle stringhe localizzabili è di fondamentale importanza. Rendere localizzabile una stringa che non deve esserlo potrebbe modificare il comportamento dell'applicazione. Al contrario, considerare non localizzabile una stringa che deve esserlo risulterà all'utente come un errore nell'interfaccia utente.

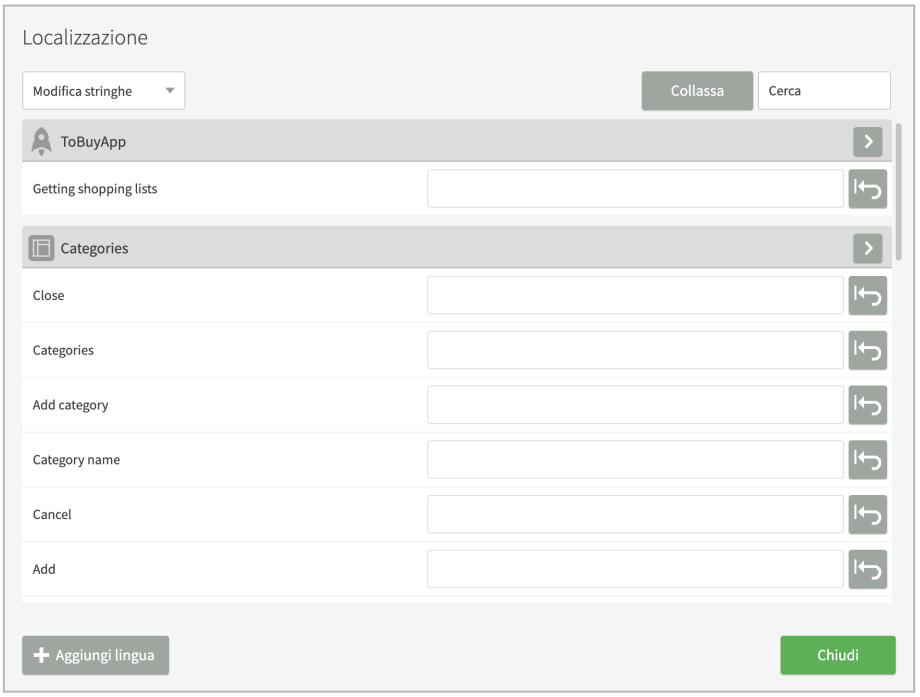
Al termine della selezione è possibile confermare le modifiche cliccando sul pulsante *Salva*. In questo momento, nell'albero del progetto verrà creata o aggiornata una nuova risorsa di tipo *Lingua* con lo stesso nome della lingua da localizzare, come mostrato nell'immagine seguente. Questa risorsa contiene i dati di localizzazione di una specifica lingua e non deve essere modificata manualmente, perché è possibile causare incoerenze nel sistema di traduzione. Se viene cancellata, verranno perse le traduzioni in una determinata lingua.
Nota bene: dopo aver confermato le modifiche, occorre salvare il progetto perché le modifiche siano effettivamente registrate.

Le stringhe non localizzabili sono consultabili selezionando la voce *Non localizzabili* nel campo di selezione in alto a sinistra. In questa lista è possibile invertire lo stato di localizzabilità per ogni stringa tramite i pulsanti sulla riga, come già visto in precedenza.



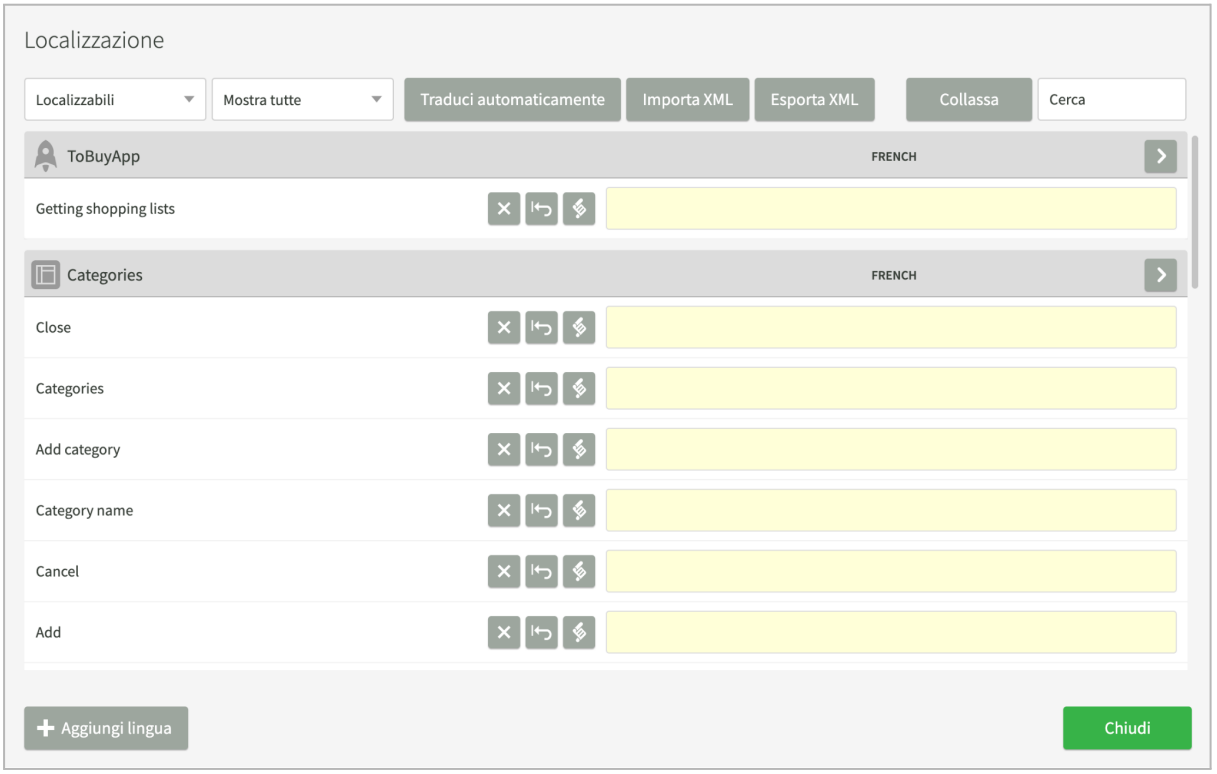
Verifica delle stringhe originali da localizzare

Prima di passare all'operazione di traduzione vera e propria, si consiglia di verificare la correttezza (ortografica, della forma e del significato) dei testi della lingua principale dell'applicazione, quella in cui è stato sviluppato il progetto, ovvero la lingua a partire dalla quale deve essere eseguita la traduzione. A tal fine è possibile selezionare la voce *Modifica originali* nel campo di selezione in alto a sinistra della pagina di localizzazione. In questa modalità di lavoro, è possibile modificare le stringhe originali. Cliccando il pulsante *Salva*, le modifiche effettuate verranno riportate nel codice sorgente o negli elementi delle videate relativi. **Nota bene**: sarà possibile modificare solo le stringhe selezionate come localizzabili.



Gestione della localizzazione

Dopo aver verificato le stringhe originali, è possibile iniziare la localizzazione vera e propria. A tal fine è possibile selezionare la voce *Localizzabili* dal campo di selezione in alto a destra nella pagina di localizzazione. Apparirà una lista come nell'esempio seguente:



Per ogni stringa è possibile inserire direttamente la versione localizzata nelle varie lingue utilizzando i campi a sfondo giallo. I pulsanti presenti su ogni riga permettono rispettivamente di:

- Rendere non localizzabile una stringa che in precedenza era stata ritenuta localizzabile o che il sistema di localizzazione ha identificato come localizzabile (icona X).
- Verificare la posizione della stringa nel progetto (icona freccia).
- Tradurre automaticamente la stringa utilizzando Google Translate (icona pergamena).

Inserendo una traduzione, essa verrà copiata in tutte le stringhe uguali successive nella lista non ancora localizzate. È **sempre** necessario verificare ogni stringa perché lo stesso testo può avere traduzioni diverse in contesti diversi.

I pulsanti in alto permettono di:

- tradurre automaticamente tutte le stringhe non ancora localizzate presenti nella lista, tenendo in considerazione anche un'eventuale ricerca in corso.
- Importare un file XML contenente delle traduzioni.
- Esportare la lista attuale in formato XML, senza tenere conto di una eventuale ricerca in corso.

L’esportazione e l’importazione dei file XML permettono di inviare un file di localizzazione ad un'agenzia esterna che può effettuare la traduzione utilizzando un software CAT. Quando le versioni localizzate vengono importate dal file XML, esse vengono marcate come definitive e appariranno nella lista con uno sfondo verde. Si considera infatti che solo le traduzioni create a seguito di un processo di localizzazione professionale possano essere considerate allo stato dell’arte, e quindi definitive.

Nella parte alta della videata è presente un secondo campo di selezione che permette di selezionare:

- tutte le stringhe localizzabili e le relative traduzioni, tramite la voce *Mostra tutte*;
- tutte le stringhe da localizzare, almeno in una lingua, tramite la voce *Mostra da localizzare*;
- tutte le stringhe non definitive tramite la voce *Mostra non importate*.

Al termine dell’operazione di localizzazione, l’applicazione sarà pronta per il test e la distribuzione nelle varie lingue supportate.

Andare a runtime

Quando l’applicazione contiene i dati di localizzazione come risorse di tipo lingua, è possibile testare la versione localizzata avviandola in anteprima nell’IDE, oppure installandola su un server di produzione.

La lingua selezionata per una determinata sessione browser viene identificata automaticamente dalla lingua del browser, oppure può essere impostata da codice nell’evento *app.onStart*, impostando la proprietà *app.langCode* al codice lingua desiderato, formato da due caratteri. La seguente riga di codice, ad esempio, mostra l’applicazione in lingua francese.

```
App.Session.prototype.onStart = function(request)
{
  app.langCode = "fr";
  …
}
```

Se la risorsa corrispondente non è presente nel progetto, verrà utilizzata quella della lingua marcata come default nella videata popup per l’aggiunta della lingua. Se nessuna lingua è stata marcata come default, verranno utilizzate le stringhe originali.

Modificando la proprietà *app.langCode* quando la sessione è già iniziata, la modifica avrà effetto solo sulle stringhe che vengono gestite dal sistema di localizzazione dopo la modifica. Se quindi viene eseguito uno script che contiene risorse localizzate, verranno utilizzate le versioni relative al nuovo valore; al contrario le videate aperte non subiranno modifiche. Se quindi si desidera modificare dinamicamente la lingua in cui viene mostrata l’applicazione, è necessario chiudere e riaprire tutte le videate, ed eventualmente ricaricare o aggiornare la visualizzazione del menu principale.

Traduzione di librerie e di applicazioni

Abbiamo visto nei paragrafi precedenti che per aprire la videata di localizzazione occorre selezionare l'oggetto applicazione nell'albero del progetto: questo consente di creare all'interno dell'applicazione le risorse di tipo lingua necessarie alla sua localizzazione. A tal fine non vengono considerati solo le videate e gli script presenti nell'applicazione, ma anche quelli presenti in tutte le librerie che essa utilizza. Le stringhe da localizzare infatti, possono essere contenute anche nelle videate o nei metodi delle classi contenuti in una libreria del progetto.

Se il progetto contiene più applicazioni, oppure se i componenti di libreria fanno parte di un componente che può essere esportato in altri progetti, può essere conveniente effettuare una localizzazione specifica di una determinata libreria. In questo modo le stringhe della libreria non verranno più considerate all'interno del processo di localizzazione delle applicazioni che la utilizzano. Per localizzare una libreria è possibile procedere esattamente come nel caso dell'applicazione, selezionando l'oggetto libreria nell'albero e poi usando il menù personalizzato *Localizzazione*. Il processo di localizzazione di una libreria permette di creare le risorse di tipo lingua all'interno della libreria stessa, e a questo punto essa verrà considerata come già localizzata quando si procede alla localizzazione delle applicazioni.

Localizzando una libreria, quindi, si evita di doverla gestire per ogni applicazione contenuta nel progetto. Se la libreria viene esportata in un componente, quando viene importata viene considerata già localizzata e quindi non entra più nel processo di localizzazione delle applicazioni che la utilizzano.

Nota bene: una libreria importata tramite un componente è un oggetto non modificabile. Se si desidera quindi modificare i dati di localizzazione è necessario sempre farlo nel progetto originale, quello da cui è stato esportato il componente.

Teamworks e gestione traduzioni

Se un progetto è gestito da un gruppo di lavoro, occorre considerare come il sistema di localizzazione interagisce con il sistema di teamworking. In particolare:

- Rendere localizzabile una stringa contenuta nel codice di un metodo e ritenuta non definita o non localizzabile modifica il codice del metodo.
- Modificare le stringhe originali modifica i metodi o le videate in cui esse sono presenti.
- Tutte le altre modifiche aggiornano le risorse di tipo lingua.

Rispetto a Teamworks, queste modifiche devono essere considerate come ogni altra modifica al progetto, quindi si applicano le operazioni di commit, push, fetch, eccetera.

Attenzione: le modifiche alle risorse di tipo lingua sono considerate come atomiche, quindi non è possibile modificarle contemporaneamente in due branch o fork diversi del progetto. I conflitti su queste risorse vengono risolti prendendo una versione o l'altra, senza unire insieme le variazioni effettuate su entrambe le versioni. È quindi necessaria un'attenta pianificazione delle operazioni di localizzazione per evitare all'origine tali conflitti.

Localizzazione di versioni successive dell’applicazione

Dopo aver completato il processo di localizzazione relativo ad una versione dell’applicazione, il percorso di implementazione della stessa normalmente procede fino ad arrivare ad una versione successiva.

A questo punto occorre considerare come cambia il processo di localizzazione della versione successiva dell’applicazione. In generale, le modifiche al progetto possono portare ai seguenti casi:

- Aggiunta di nuove stringhe da localizzare o non definite.
- Eliminazione di stringhe già localizzate.
- Modifica di stringhe già localizzate.

Le operazioni di localizzazione di una versione successiva dell’applicazione avvengono con gli stessi strumenti visti finora. In particolare:

- La lista delle stringhe non definite conterrà tutte e sole le nuove stringhe da localizzare, che non siano state gestite nel codice tramite la funzione *t*.
- I dati di localizzazione delle stringhe non più presenti nel progetto verranno eliminati al primo aggiornamento delle risorse di localizzazione.
- Le stringhe modificate dovranno essere nuovamente localizzate. Non vengono mantenuti i dati di localizzazione precedenti. Se si utilizza un processo di localizzazione professionale, ovvero le stringhe vengono sempre esportate in un file XML per essere localizzate in un software CAT, sarà lo stesso software CAT a rilevare le modifiche. In questo modo il lavoro fatto sulle traduzioni precedenti delle stringhe modificate può essere parzialmente recuperato da chi utilizza il software CAT. Ciò, nel lungo termine, genera notevoli vantaggi a livello di tempi e di costi rispetto al ricorso alla traduzione manuale o automatica delle stringhe nel sistema di traduzione di Instant Developer Cloud.

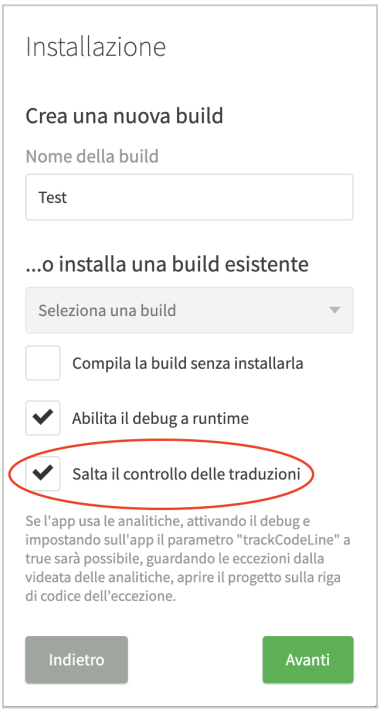
Nella videata di gestione della localizzazione è possibile utilizzare il secondo campo di selezione per vedere la lista delle sole stringhe da localizzare. In questo modo è possibile intervenire solo sulla parte di progetto che è variata. Per quanto riguarda l’esportazione di file XML per le agenzie di traduzione, è possibile scegliere se esportare tutte le stringhe localizzabili, oppure solo quelle ancora da localizzare, a seconda del metodo di localizzazione desiderato dall’agenzia.

Errori di installazione relativi alla localizzazione

Durante l’installazione di un’applicazione in un server di produzione, è possibile ricevere un errore relativo alla mancanza dei dati di localizzazione dell’applicazione. Quando infatti la console chiede al server IDE di compilare una build dell’applicazione, se essa contiene almeno una risorsa di tipo lingua si attiva un algoritmo di controllo che verifica se tutte le stringhe localizzabili sono effettivamente state localizzate e che non siano presenti stringhe non definite.

In mancanza di una localizzazione completa, la build fallisce e non è possibile installarla. Per risolvere questo problema occorre aprire il progetto e inserire i dati di localizzazione, anche utilizzando il sistema di traduzione automatica.

È possibile saltare il controllo della localizzazione in fase di build abilitando il flag *Salta il controllo delle traduzioni* nella videata delle opzioni di traduzione, che però è attivo solo se si abilita anche l'opzione di debug a runtime, in modo da marcare la build come di test e non come installazione definitiva.



Funzionamento del framework di localizzazione

Il framework di localizzazione è incluso nel runtime dell'applicazione e utilizza i dati delle risorse linguistiche per tradurre nella lingua giusta l'interfaccia utente di ogni sessione. Vediamone ora il funzionamento.

Compilazione delle risorse linguistiche

Le risorse di tipo lingua vengono considerate nella fase di compilazione dell'applicazione per preparare un oggetto JSON definito a livello di applicazione (App), in modo da essere caricato in memoria una sola volta per tutte le sessioni di un processo worker. Questo oggetto JSON viene espresso nel codice dell'applicazione come esemplificato di seguito:

```
App.i18m = {
  "it": {
    "5B9RRmi7su9zRw7PEh1WUg==&backButtonText": "Chiudi",
    "46CDblbeDDGEnjoCIgY+vQ==&innerText": "Categorie",
    "hS6xEjWvuLTr0BcVuguCew==": "Aggiungi categoria",
    "5X3SL657a3vC4CjsKOOnBA==": "Nome della categoria",
```

```
      ...
   }
}
```

Per ogni lingua vengono quindi riportate le sole informazioni relative alle stringhe traducibili, identificate da un identificativo dell'oggetto del progetto a cui esse si riferiscono e dal nome della proprietà che rappresentano.

Localizzazione delle stringhe nel codice

Queste informazioni vengono utilizzate dalla funzione *t,* definita nella classe del framework *App.Utils*, che ha due parametri: *text* e *params*.

Normalmente *text* è una stringa e *params* contiene i valori da sostituire. Quando è attivo il sistema di localizzazione, il compilatore converte il primo parametro da stringa ad oggetto, passando alla funzione l'ID della stringa che corrisponde all'ID del token corrispondente calcolato dal parser di codice di Instant Developer, il codice del *locale* da utilizzare ed infine il testo passato come parametro nello script del progetto. Vediamo un esempio di come il codice viene trasformato.

```
// Codice scritto nel progetto
let msg =  t("Getting @num shopping lists", {num:12});

// Codice trasformato (se è attivo il framework di localizzazione)
let msg = t({id:"UfceiHW+9AfQQ02DH4tp6Q==", lc: app.langCode, t: "Getting
@num shopping lists"}, {num:12});
```

A questo punto la funzione *t* ha tutte le informazioni necessarie per puntare nella mappa delle risorse alla versione localizzata della stringa passata come parametro.

Localizzazione degli elementi visuali e delle liste valori

La localizzazione degli elementi visuali e delle liste valori avviene sempre sfruttando la funzione *t.* Quando l'applicazione è localizzata, il compilatore che si occupa di generare il codice per l'inizializzazione delle videate e delle liste valori non scrive direttamente il valore delle proprietà traducibili, ma genera una chiamata alla funzione *t* in modo da ottenere a runtime il valore localizzato di tali costanti.

Localizzazione di numeri e date

Oltre alla gestione delle stringhe dell'interfaccia utente, il processo di localizzazione comprende la gestione dei formati numerici, monetari e di data.

Instant Developer Cloud non include un framework automatico per questo tipo di operazioni, ma si basa sulle funzioni disponibili negli ambienti JavaScript. In particolare, per l'espressione delle date vengono usate le librerie moment e moment-timezone che consentono la conversione di date in stringhe e viceversa in ogni *locale*.

Per quanto riguarda i formati numerici, JavaScript fornisce il supporto completo con la funzione toLocaleString ed equivalenti.

L'oggetto app.locale

Oltre alla proprietà *app.langCode*, Il framework di Instant Developer contiene l'oggetto *app.locale* che contiene alcune proprietà e metodi utili a questo scopo:

- *app.locale.timeZone*: nome del timezone della sessione.
- *app.locale.timeFormat*: formato ora richiesto dalla sessione.
- *app.locale.dateFormat*: formato data richiesto dalla sessione.
- *app.locale.moment(input)*: metodo che restituisce un oggetto *moment* per la manipolazione delle date.
- *app.locale.now()*: metodo che restituisce un oggetto *moment* che rappresenta data e ora attuali.
- *app.locale.today()*: metodo che restituisce un oggetto *moment* che rappresenta la data di oggi.
- *app.locale.time()*: metodo che restituisce un oggetto *moment* che rappresenta l'orario attuale.

Per ottenere quindi la formattazione corretta di un campo *datetime* proveniente dal database, è possibile utilizzare la seguente espressione.

```
let datetime = …;
let text = app.locale.moment(datetime).format("L");
```

Il metodo *app.locale.moment* è quindi il punto di accesso per l'utilizzo della libreria moment.js nei propri progetti. Una volta ottenuto l'oggetto *moment* che rappresenta un valore *datetime*, sono applicabili tutti i metodi della relativa libreria.

Input di valori numerici e date

Gli elementi visuali che consentono l'input dell'utente di valori numerici e date variano in funzione del framework grafico utilizzato. Se si utilizza IonicUI, il framework consigliato, si hanno a disposizione i seguenti elementi:

- *IonInput (type:number)* per l'inserimento di numeri.
- *IonDateTime* per l'inserimento di valori *date*, *time* e *datetime*.

Nel caso di utilizzo di *IonDateTime*, si ricorda di specificare le proprietà *displayFormat* e *pickerFormat* usando i valori delle proprietà *app.locale.dateFormat* e *app.locale.timeFormat*.

Dalla versione 22, è possibile anche utilizzare input con visualizzazione ed inserimento mascherato sia di valori numerici che di date. Si consiglia di leggere la documentazione della proprietà *mask* per ulteriori informazioni.