MIT APP INVENTOR 2

di FRANCESCO FICILI

Continuiamo il nostro viaggio alla scoperta di MIT App Inventor, entrando nel merito della programmazione grafica e illustrando alcune delle proprietà più interessanti di questo eccezionale tool per lo sviluppo di app Android. Seconda Puntata.

Nella prima puntata di questo corso abbiamo introdotto MIT App Inventor (abbreviato di seguito con AI), l'innovativo ambiente di sviluppo grafico per applicazioni Android creato da Google e sviluppato attualmente dal MIT. Abbiamo presentato l'ambiente e l'emulatore, approfittando per presentare un primo, semplice, esempio. In questa puntata entriamo maggiormente nel merito della programmazione grafica, illustrando alcune delle proprietà più interessanti di AI.

Programmazione in MIT App Invertor

Sapete ormai che in AI la programmazione avviene completamente tramite l'uso di elementi grafici.

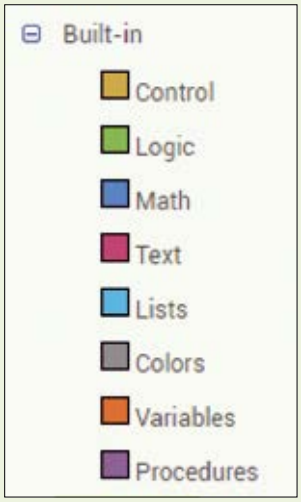

Fig. 1 - Insieme dei blocchi built-in.

Fig. 2 - Esempio di blocchi del gruppo control.

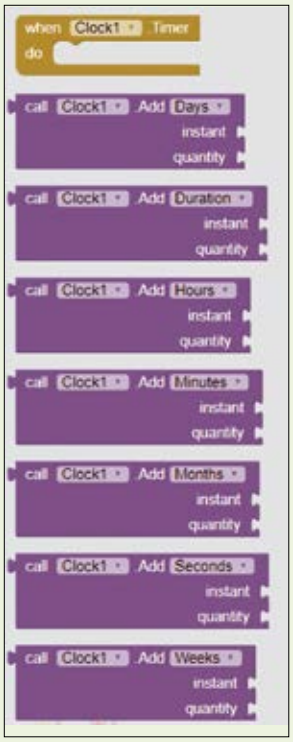

Fig. 3 - Esempio di blocchi del componente clock.

Tali elementi in AI sono generalmente chiamati "Behaviours" (comportamenti) e prendono la forma di blocchi simili alle tessere di un puzzle (nel resto della puntata e nelle successive utilizzeremo il termine blocchi e behaviours indistintamente, riferendoci sempre a questi elementi). I blocchi in AI possono appartenere fondamentalmente a due gruppi:

- built-in; si tratta dei blocchi base, sempre presenti a prescindere dai componenti inseriti in una determinata app (in **Fig. 1** è riportato l'insieme dei blocchi built-in e in **Fig. 2** alcuni blocchi del gruppo control);
- componenti; si tratta dei blocchi specifici per i componenti; ne esiste una grande varietà, in quando questi blocchi cambiano da componente a componente, ma si possono identificare delle classificazioni (in **Fig. 3** è riportato un esempio per il componente clock).

Beige	Strutture di controllo
Verde	Blocchi logici
Blu	Blocchi matematici
Fucsia	Blocchi di testo (stringhe)
Celeste	Blocchi gestione Liste
Grigio	Blocchi gestione colori
Arancio	Blocchi gestione variabili
Viola	Blocchi controllo procedure

Tabella 1 - Significato dei colori per i componenti built-in.

Beige	Eventi
Fucsia	Metodi
Verde Scuro	Setter di proprietà
Verde Chiaro	Getter di proprietà

Tabella 2 - Significato dei colori per i behaviours più comuni dei componenti.

In App Inventor, forma e colore di un behaviour hanno un loro significato; in particolare si usa, per i blocchi built-in, l'insieme di associazioni riportato in **Tabella 1**, mentre per quanto riguarda i behaviour dei componenti, i più comuni significati dei colori sono riportati in **Tabella 2**.
Inoltre, sostanzialmente tutti i blocchi presentano uno o più connettori di ingresso e di uscita, sagomati come rientranze o sporgenze, tipo tasselli di puzzle. In prossimità del connettore è solitamente presente una label che serve ad identificare l'input o l'output del connettore stesso.
Abbiamo iniziato ad anticipare dei concetti parlando di eventi, metodi, setter e getter, che saranno chiariti nei paragrafi successivi, quando parleremo dei componenti. Prima di arrivare a quel punto però, occorre illustrare come AI gestisce due aspetti che sono alla base di qualsiasi linguaggio di programmazione: le strutture dati e le strutture di controllo.

Variabili

Uno degli aspetti fondamentali di un qualsiasi linguaggio di programmazione è la gestione delle strutture dati, ossia le tipologie di dati supportate e le operazioni che possono essere eseguite su di esse. AI prevede una gestione estremamente semplificata delle strutture dati: si possono dichiarare variabili

Fig. 4 - Creazione e inizializzazione di una variabile globale.

Fig. 5 - Blocchi di inizializzazione di variabili local in AI.

globali e locali, mentre sono supportati i seguenti tipi di dati:

- numerici;
- booleani;
- stringhe;
- liste:
- colori.

Vediamo come definire variabili globali e locali tramite un semplice esempio.
Per definire una variabile globale è sufficiente posizionare il blocco "initialize global name to" del gruppo built-in "Variables" in un punto qualsiasi del nostro Block Viewer e rinominare la sezione "name" con il nome della nostra variabile (nel nostro esempio è *MyVar*), come illustrato in **Fig. 4**. In questo caso viene creata una variabile globale denominata MyVar ed inizializzata al valore numerico '0'. Tale variabile sarà visibile ovunque all'interno della nostra app.
Le variabili locali invece hanno una visibilità (o *scope*) limitato ad una specifica procedura; tale tipologia di variabile può essere definito tramite due blocchi distinti in AI, come illustrato in **Fig. 5**.
La differenza sta solo nel fatto che il primo blocco può essere utilizzato nella sezione **do** di un blocco superiore (tipicamente un evento), mentre il secondo nella sezione **return** di un blocco collegato: ad esempio un blocco di set (come abbiamo visto in precedenza i blocchi in AI sono sagomati in modo da incastrarsi tra loro solo in un determinato modo, così da garantire intrinsecamente la compatibilità).
In **Fig. 6** è illustrato un esempio che mostra l'utilizzo dei due blocchi per settare il valore di una label; nell'esempio vengono utilizzati due eventi (nello specifico *Screen1.Initialize* e *Screen1.ScreenOrientationChanged*) per settare il valore della label Label1 al valore corrispondente, rispettivamente, alle variabili locali MyLocal1 e MyLocal2, recuperati tramite l'uso di un blocco "get".
Nell'esempio appena visto abbiamo introdotto anche una delle due operazioni fondamentali che possono essere eseguite su una variabile, ossia l'operazione di **get** e quella, altrettanto importante, di **set**, che si utilizza quando vogliamo assegnare un valore ad una variabile. È importante notare che ogni volta che viene creata una variabile, locale o globale, se ci posizioniamo con il puntatore del mouse sopra il nome della variabile stessa, compare sullo schermo una shortcut che ci permette di accedere velocemente ai blocchi di get e set per quella variabile, come si vede in **Fig. 7**.

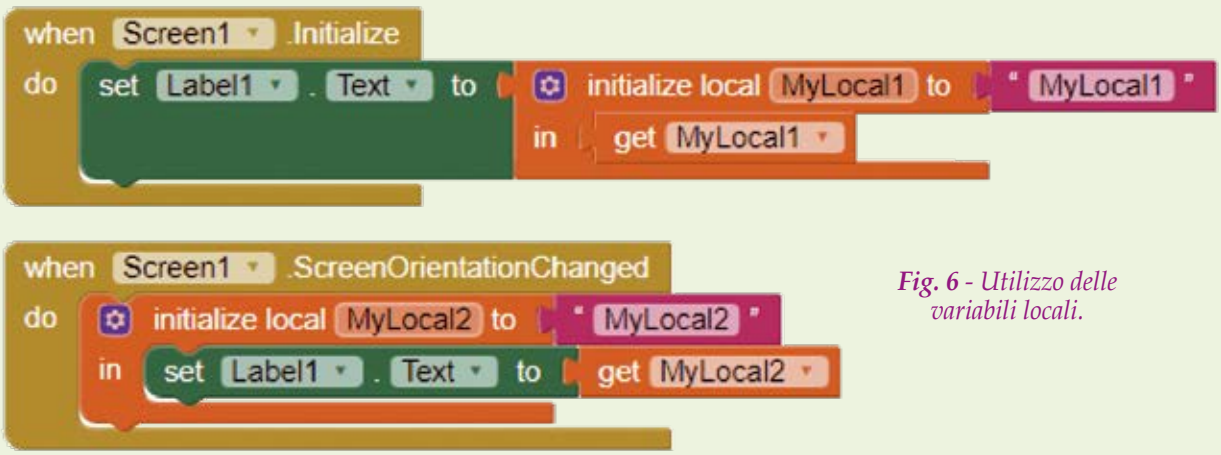

Fig. 6 - Utilizzo delle variabili locali.

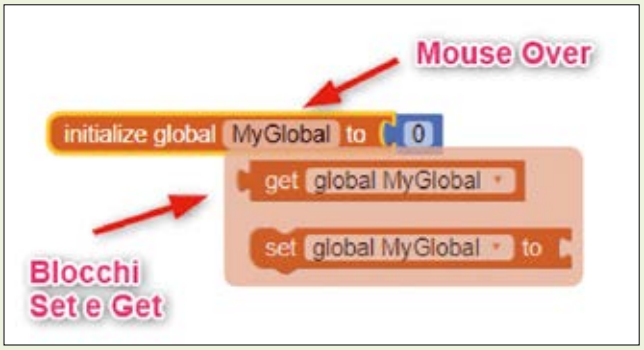

Fig. 7 - Blocchi Set e Get per la variabile MyGlobal.

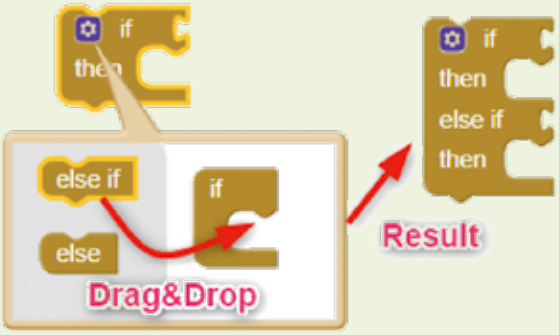

Fig. 9 - Modifica di un blocco if.. then..else if.

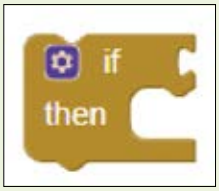

Fig. 8
Blocco if..then base.

Strutture di controllo

Passiamo ora all'analisi dell'altro aspetto fondamentale del linguaggio dopo le strutture dati, ossia le strutture di controllo. AI presenta le strutture di controllo classiche che possono essere trovate nei più comuni linguaggi, più ulteriori strutture espressamente pensate per la programmazione di applicazioni in ambiente android. AI racchiude tutte queste strutture all'interno del gruppo di behaviours built-in denominato "control"; essenzialmente si possono identificare tre gruppi di strutture di controllo:

- strutture di controllo del flusso di esecuzione;
- loop;
- strutture di controllo dell'applicazione.

Controllo di Flusso

Le strutture di controllo di flusso consentono di eseguire operazioni di tipo decisionale, per mezzo di blocchi condizionali come i blocchi "if..then" e "if.. then..else". Tali blocchi operano su risultati di operazioni booleane o di confronto. Il blocco base è il blocco "if…then", rappresentato nella sua forma base in **Fig. 8**, che data una condizione iniziale, da collegare sull'ingresso "if" la verifica e, in caso sia vera esegue il ramo "then", altrimenti la salta.
Tale blocco può essere modificato cliccando sul pulsante a forma di ingranaggio, aggiungendo ulteriori branch, come illustrato in **Fig. 9**, dove un blocco if… then base viene modificato in una condizione doppia if..then..else if. Rispetto al caso precedente, in questo frangente viene valutata la condizione relativa all'ingresso "if", se quest'ultima è vera, viene eseguito il ramo "then", altrimenti viene saltata e si passa alla verifica della condizione del ramo "else if". Su un blocco if..then può essere inserito un numero arbitrario di rami "else..if", ma solo un ramo "else", che identifica il ramo da eseguire nel caso nessuna delle condizioni precedenti sia stata verificata.

Loop

I loop (o cicli) sono blocchi pensati per l'esecuzione di operazioni iterative, sia a conteggio (cicli for) che a condizione d'uscita (cicli while). Partiamo dai cicli a conteggio: AI mette a disposizione due tipi di ciclo "for": uno generico ed uno specifico per la gestione delle liste, entrambi illustrati in **Fig. 10**.
Il primo è il classico ciclo a conteggio, che definisce al suo interno una variabile locale number ed esegue il blocco di codice collegato sul connettore "do" per valori della variabile number compresi tra gli estremi "from" e "to", incrementando a ogni iterazione number di una quantità pari a "by". I valori di "from", "to" e "by" possono essere costanti oppure calcolati tramite una opportuna espressione matematica. Invece il blocco "for each item in list" esegue semplicemente un numero di iterazioni pari al numero di elementi presenti nella lista che viene collegata al suo connettore "list". Per quanto riguarda invece i cicli a condizione di uscita AI rende disponibile un unico ciclo "while", rappresentato in **Fig. 11**.

Il funzionamento è semplicissimo, il blocco valuta ad ogni iterazione la condizione collegata al terminale "test", se questa è verificata esegue il codice grafico collegato al connettore "do", altrimenti termina. Chiaramente, anche in questo caso la condizione può essere una costante o un'espressione complessa a piacere; l'importante è che il risultato alla fine sia un valore booleano.

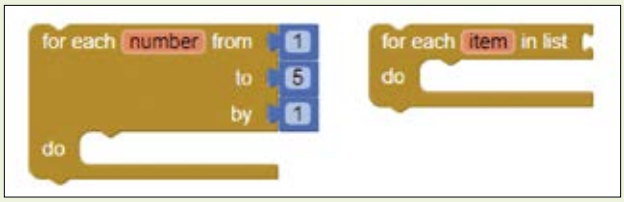

Fig. 10 - Tipologie di cicli for supportate da AI.

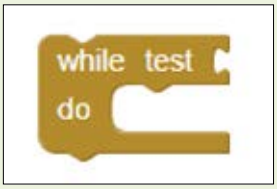

Fig. 11 - Ciclo While.

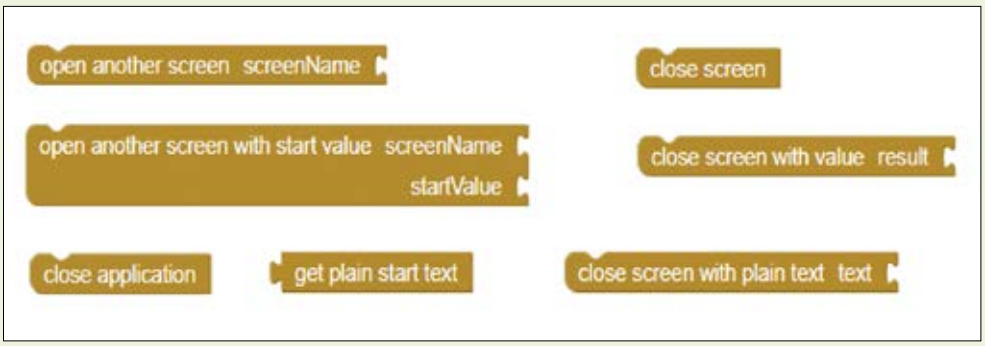

Fig. 12 - Strutture di controllo dell'applicazione.

Controllo dell'applicazione

Veniamo infine ai blocchi di controllo dell'applicazione; AI mette a disposizione diverse strutture di controllo di questo tipo, in **Fig. 12** ne abbiamo riportate alcune.
Tali strutture di controllo permettono di gestire ad alto livello la nostra app, ad esempio ci consentono di passare da uno screen ad un altro in app multi-screen, chiudere o aprire screen e chiudere l'app stessa.

I componenti in App Inventor

Ora che abbiamo illustrato strutture dati e strutture di controllo, possiamo spiegare cosa si intende in AI per componenti. I componenti (o components) non sono altro che gli elementi che ci consentono di aggiungere funzionalità alla nostra app. Come abbiamo visto nella prima puntata, possiamo inserirli nella "Designer View" trascinandoli dalla "Components Palette" all'interno della "Designer View", dove è rappresentato il layout della nostra app. I componenti possono essere visibili o meno, a seconda del fatto che abbiamo o meno una componente grafica; ad esempio un componente button è visibile sul layout in quanto ha sia una componente grafica (l'immagine del pulsante stesso) che una serie di blocchi associati. Un componente Accelerometer, invece, è non visibile, in quanto non ha parte grafica ma solo una serie di blocchi associati al funzionamento nella "Block View".
I componenti non visibili vengono comunque visualizzati come icone nella parte bassa della "Designer View", in modo da poter essere comunque selezionati e da poter accedere alle loro proprietà.
Prima di proseguire facciamo una veloce rassegna dei componenti principali di Ap Inventor.
User Interface: si tratta della famiglia di componenti più estesa, tramite la quale è possibile creare l'interfaccia grafica della nostra app. A questa famiglia appartengono i seguenti elementi:

- Button; il classico pulsante grafico;
- CheckBox; box di scelta true/false;
- DatePicker; picker che permette di selezionare una data tramite finestra di pop-up;
- Image; permette di mostrare un'immagine sul layout della nostra app;
- Label; consente di stampare del testo a video;
- ListPicker; picker che permette di selezionare un elemento da una lista;
- ListView; permette di visualizzare gli elementi di una lista;
- Notifier; permette di visualizzare dei messaggi all'utente;
- PasswordTextBox; elemento che permette di inserire un input testuale in modalità password (invece dei caratteri alfanumerici vengono stampati degli asterischi);
- Screen; serve a inserire schermate aggiuntive;
- Slider; elemento che permette all'utente di fornire un input numerico muovendo il cursore di una slide bar;
- Spinner; elemento che permette di creare un drop-down menu;
- TetxBox; elemento che permette di inserire un input testuale;
- TimePicker; picker che permette di selezionare un'ora tramite finestra di pop-up;
- WebView; elemento che permette di aprire e visualizzare una URL.

Layout: si tratta di una famiglia di componenti che permettono di organizzare il layout della nostra app, creando una sorta di griglia che permette di collocare gli elementi grafici sul layout. Ci sono sostanzialmente tre possibili scelte per il layout:

- VerticalArrangement; è layout verticale;
- HorizontalArrangement; layout orizzontale;
- TableArrangement; è il layout tabellare.

Media: questa famiglia di componenti consente di gestire elementi multimediali all'interno della nostra app. All'interno di questa famiglia troviamo:

- Camcorder; è il componente che permette di aprire la videocamera del dispositivo per effettuare la registrazione di un video;
- Camera; permette di aprire la videocamera del dispositivo per scattare una foto;
- ImagePicker; è il componente che consente di selezionare un'immagine tra quelle presenti nella galleria del device;
- Player; permette di riprodurre un file audio e di controllare la vibrazione del device (per file audio di lunga durata);
- Sound; consente di riprodurre un file audio e di controllare la vibrazione del device (per file audio di breve durata);
- SoundRecorder; permette di accedere al microfono del dispositivo ed effettuare una registrazione audio;
- SpeechRecognizer; permette di attivare la funzionalità di riconoscimento vocale integrata in Android al fine di convertire un parlato in testo;
- TextToSpeech; permette di trasformare un testo in un parlato attraverso un sintetizzatore vocale (supporto multilingua);
- VideoPlayer; permette di riprodurre un file video all'interno di un player dotato dei normali comandi attivabili tramite touch screen;
- YandexTranslate; permette di effettuare traduzioni in tempo reale attraverso le API offerte dal traduttore automatico di Yandex.

Drawing & Animation: questa famiglia offre una serie di componenti per gestire animazioni grafiche e disegni. L'elemento base è il "canvas", ossia una sorta di blocco all'interno del quale è possibile effettuare disegni ed animazioni. Questa famiglia comprende i seguenti componenti:

- Canvas; è un blocco rettangolare all'interno del quale è possibile disegnare o effettuare delle animazioni;
- Ball; sprite circolare che può essere posizionato all'interno di un canvas, dove può interagire con altri sprites e reagire al touch;
- ImageSprite; sprite che può essere associato ad un'immagine e posizionato all'interno di un canvas, dove può interagire con altri sprite e reagire al touch.

Maps: famiglia di component specifici per la gestione di mappe. All'interno di questa famiglia troviamo:

- Circle; è il componente che permette di visualizzare un cerchio di un certo raggio (in metri) ad una determinata latitudine e longitudine;
- FeatureCollection; fornisce un contenitore per caricare caratteristiche multiple quali gruppo e gruppo di poligoni rappresentanti gli stati degli United States (ciascuna caratteristica diviene un proprio componente che può essere editato nel designer);
- LineString; componente che permette di disegnare una sequenza di linee su una mappa;
- Map; componente che esegue il rendering di mappe e permette di posizionare marker per identificare punti sulla mappa renderizzata;
- Marker; componente che permette di posizionare un marker su una mappa;
- Polygon; componente che permette di disegnare un poligono su una mappa;
- Rectangle; è il componente che permette di disegnare un rettangolo su una mappa.

Sensors: famiglia di componenti che permette di accedere ai sensori interni del dispositivo. Tra essi troviamo:

- Accelerometer; componente che permette di interfacciare l'accelerometro interno;
- BarcodeScanner; componente che permette di utilizzare la camera come scanner per codici a barre;
- Clock; componente che permette di generare eventi periodici;
- Gyroscope; componente che permette di interfacciare il giroscopio interno;
- LocationSensor; componente che permette di effettuare operazioni di geolocalizzazione;
- NearField; componente che permette di gestire l'interfaccia NFC del dispositivo;
- OrientationSensor; componente che permette di interfacciare la bussola elettronica interna;
- Pedomter; componente che consente, tramite l'accelerometro interno, di contare i passi;
- Proximity; è il componente che permette di rilevare la distanza di un oggetto (in cm) rispetto allo schermo.

Social: famiglia di componenti specifica per la gestione di funzionalità social. I componenti che fanno parte di questa famiglia sono:

- ContactPicker; picker che permette di selezionare un contatto dalla rubrica;
- EmailPicker; picker che permette di trovare un indirizzo email inserendo il nome del contatto;
- PhoneCall; componente che consente di effettuare una chiamata telefonica al numero specificato;
- PhoneNumberPicker; picker che permette di ricavare il numero di telefono inserendo il nome del contatto;
- Sharing; componente che permette di effettuare sharing (di file o messaggi) tra la app ed altre app installate sul dispositivo;

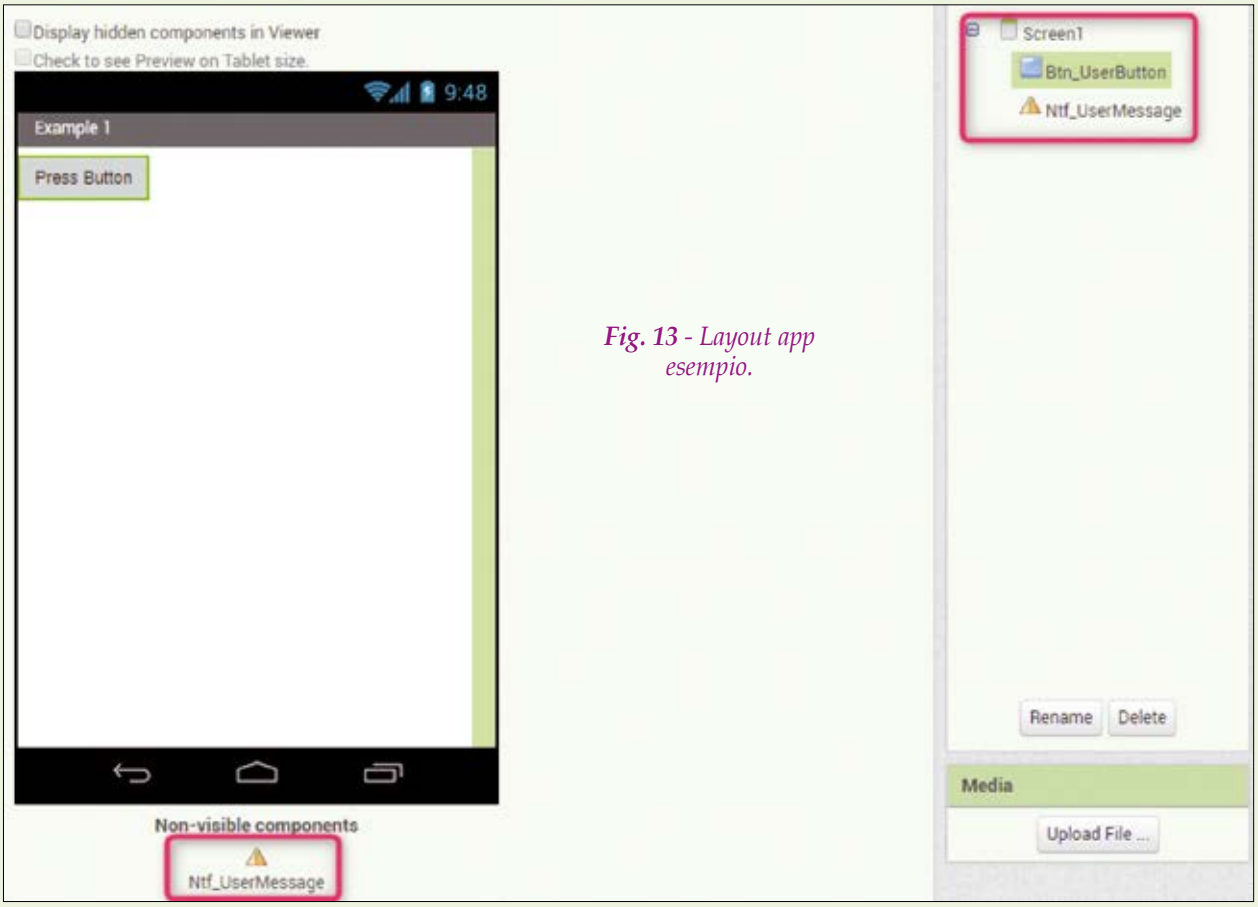

Fig. 13 - Layout app esempio.

- Texting; componente che permette di inviare e ricevere SMS;
- Twitter: componente che permette di interagire con twitter.

Storage: famiglia di componenti che permette la gestione della memorizzazione dei dati all'interno del dispositivo. Questa famiglia comprende i seguenti componenti:

- File; componente che permette di leggere e scrivere file;
- TinyDB; componente che permette di gestire un piccolo DB sul dispositivo;
- TinyWeDB; componente che permette di accedere ad un web service per immagazzinare e recuperare informazioni.

Connectivity: famiglia di componenti per la gestione della connettività. Questa famiglia comprende i seguenti componenti:

- ActivityStarter; componente che permette di lanciare una activity;
- BTClient; il componente che permette di gestire un client Bluetooth;
- BTServer; componente che permette di gestire un server Bluetooth;
- Web; componente che permette di gestire metodi HTTP GET, POST, PUT e DELETE.

Informazioni più dettagliate sui componenti possono essere trovate sul sito ufficiale di AI, all'indirizzo:
http://ai2.appinventor.mit.edu/reference/components/.

Esempio pratico: utilizzo del componente notifier

Ora che abbiamo completato anche il nostro excursus sui componenti in MIT AI, passiamo, come di consueto al nostro esempio pratico, sfruttando alcune delle conoscenze acquisite. Ci proponiamo di realizzare un'app molto semplice che, alla pressione di un tasto, visualizzi un messaggio all'utente.
Creiamo quindi un nuovo progetto, come visto nella precedente puntata e iniziamo ad collocare sul layout i componenti che ci occorrono.
Per realizzare quanto abbiamo sopra indicato ci servono i seguenti componenti:

- un componente Button;
- un componente Notifier.

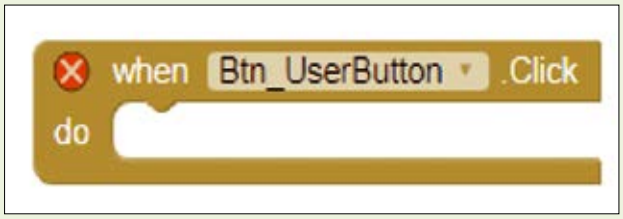

Fig. 14 - Blocco When Btn_UserButton.Click.

Fig. 15 - Blocco Call Ntf_UserMessage.ShowMessageDialog.

Fig. 16 - Codice grafico dell'esempio pratico.

Selezioniamoli dalla "Compoents Palette" (si trovano entrambi sotto il gruppo "User Interface"), posizioniamoli sul layout e poi cambiamone i nomi, rispettivamente in Btn_UserButton e Ntf_UserMessage. Inoltre cambiamo la proprietà Text del pulsante in "Press Button".
Una volta che avrete fatto ciò dovreste ottenere un risultato simile a quello di **Fig. 13**.
Passiamo a questo punto sulla sezione "Blocks" e scriviamo il codice grafico per l'implementazione della nostra app. Per realizzare quanto detto in precedenza, selezioniamo il blocco When Btn_UserButton.Click, presente tra i blocchi del componente button, ed il blocco Call Ntf_UserMessage.ShowMessageDialog, tra i blocchi del componente notifier (**Fig. 14** e **Fig. 15**) e trasciniamoli all'interno dell'area di lavoro.
A questo punto inseriamo il blocco di notifica all'interno dell'evento Click ed aggiungiamo tre costanti stringa (Text → Text String) connesse ai tre connettori del blocco di notifica, inserendo le stringhe indicate in **Fig. 16**.
In questo modo alla pressione del tasto, il componente notifier presenterà un pop-up message dal titolo "message" che mostra la stringa "You pressed the button!!!" e contenente un pulsante di conferma di lettura "OK".
Il risultato che si ottiene con l'emulatore è visibile nella **Fig. 17**.

Conclusioni

Siamo così arrivati al termine della seconda puntata, nella quale abbiamo approfondito i dettagli della programmazione grafica, illustrando strutture e dati e strutture di controllo di App Inventor e dato un primo sguardo d'insieme ai componenti, oltre ad aver presentato il solito progetto pratico.
Dalla prossima puntata inizieremo una serie di approfondimenti sui vari componenti presenti in MIT App Inventor, corredata da diversi esempi pratici di utilizzo.

Fig. 17 Risultato ottenuto tramite l'uso dell'emulatore.