PASSEPARTOUT®
Software e servizi gestionali

Corso Sprix Mobile

Gianluca Suzzi

PASSEPARTOUT®
Software e servizi gestionali

OBBIETTIVO DEL CORSO

Il corso intende entrare più nel dettaglio su alcuni aspetti della programmazione di App Mobile mediante l'utilizzo del linguaggio Sprix-Mobile: vedremo come gestire casi particolari e come utilizzare le nuove funzionalità introdotte con le ultime versioni.

Prerequisiti necessari per il corso:

- ***conoscenza di base del linguaggio SPRIX***
- ***corso di programmazione base Sprix Mobile (Marzo 2020)***

Per tutti i dettagli sull'utilizzo delle istruzioni fare riferimento al manuale SprixMobile scaricabile dalla sezione Area Sviluppatori della vostra Area Riservata sul portale di Passepartout.

La Passapp utilizzata per il seguente corso è scaricabile dall'area condivisa per i partner a questo indirizzo https://download.passepartout.cloud/CondivisaPartner/PassMobile/

PROGRAMMA CORSO

- Argomenti:
  - Navigare tra diversi Form
  - Approfondimenti sulle chiamate remote (SRVCALL e Collage Server Remoto)
  - Campi obbligatori e gestione degli errori
  - Gestire layout delle liste
  - Lettura e scrittura di parametri su dispositivo mobile (GETLOC,PUTLOC)
  - Utilizzo del GPS
  - Come personalizzare App scritte da terzi (Estensioni Sprix-Mobile e Collage-Mobile)

NAVIGARE TRA FORM DIFFERENTI

La visualizzazione di ulteriori form oltre al padre, avviene richiamando l'istruzione WSHOWFORM,<ID_FORM>, dove ID_FORM è l'id oggetto del form che vogliamo mandare a video.

Bisogna però prestare attenzione a come gestire la navigazione tra il form padre (il primo) e i figli o tra un form figlio e i fratelli evitando errori che possono bloccare l'esecuzione. La regola generale è che quando si fa un *back* e si torna al form precedente, il form attuale viene distrutto.

CASO1:

Apertura di un form partendo dal padre: in questo caso il form figlio dovrà essere creato ogni volta e mandato a video. Questo perché, quando con il *back* da dispositivo mobile o con la freccia ← da Passmobile, si torna al form padre, il form figlio viene distrutto e deve quindi essere ricreato. (vedi esempio Form padre/figlio)

NAVIGARE TRA FORM DIFFERENTI

CASO2:

Apertura di un form figlio partendo da un figlio (padre -> figlio 1-> figlio2): in questo caso quando si torna al form figlio 1 da figlio 2 non si può ricreare il form perché il form figlio 1 è ancora esistente. L'unico modo per tornare al figlio 1 è tramite pulsante *back* del dispositivo mobile o da programma tramite WSHOWFORM. Nel caso invece si passi da figlio 1 a figlio 2 il form va sempre ricreato. (Vedi esempio Form figlio/figlio)

CASO3:

Modificare il form padre a seguito di operazioni eseguite da un figlio o dal padre stesso: nel caso un'operazione richieda di tornare al form padre mostrando oggetti diversi, è necessario ricreare il form da zero (es. aggiungere una casella di testo che prima non era presente). In questo caso, dato che il form padre non viene mai distrutto (a meno che non si cambi menu), è necessario forzare la sua chiusura con l'istruzione WCALL "CLOSEFORM",<ID_FORM>.

In questo modo il form potrà essere ricreato con le modifiche necessarie.

(Vedi esempio Modifica padre da figlio)

NAVIGARE TRA FORM DIFFERENTI

In tutti i casi proposti, ogni qualvolta si abbandona un form per tornare al precedente (es. da figlio a padre) oppure per cambiare voce di menu (es. da padre ad altro sprix mobile), è possibile intercettare l'evento di chiusura in modo tale che possano essere adottate opportune politiche di avviso o salvataggio dei dati.

La chiusura del form scatena 2 eventi:

- ON_PRECLOSE: scatta prima della chiusura per dare la possibilità all'utente di scegliere se rimanere o meno sul form. In questo frangente è possibile visualizzare un messaggio valorizzando la variabile _WFPCFMSG$
- ON_CLOSE: scatta quando il form viene chiuso

(vedi esempio Modifica con avviso)

CHIAMATE REMOTE

Le chiamate remote vengono sempre scatenate da client (dispositivo mobile) verso server tramite l'istruzione SRVCALL "<nome_col_rem>","<nome_etichetta>","<call_back>"

I dati sono scambiati tramite file JSON che viene prodotto in maniera trasparente utilizzando le seguenti istruzioni, sia lato cliente che lato server:

- Istruzioni per scrivere:
  - PUTREM_STR "<chiave>","<valore>"
  - PUTREM_NUM "<chiave>",<valore>
  - PUTREM_ARRAY "<chiave>","<valore>"
  - PUTREM_FILE "<chiave>","<valore>"
- Istruzioni per leggere:
  - GETREM_STR "<chiave>","<valore>"
  - GETREM_NUM "<chiave>",<valore>
  - GETREM_ARRAY "<chiave>","<valore>"
  - GETREM_FILE "<chiave>","<valore>"

CHIAMATE REMOTE

I concetti fondamentali da tenere presenti quando si opera con le chiamate remote sono sostanzialmente 2:

1. si tratta di operazioni asincrone. Questo vuol dire che l'esito della chiamata remota, sia esso negativo (es. per errori di comunicazione col server) che positivo, è possibile conoscerlo **solo ed esclusivamente** quando scatta la call back.
   Tutta la gestione degli errori e l'elaborazione dei dati ricevuti deve quindi essere fatta dentro l'etichetta di call back.
2. Sono operazioni che presuppongono una connessione permanente con il server (il dispositivo deve quindi essere online) e in caso di trasferimento di volumi notevoli di dati (file JSON di diversi MB) potrebbe rallentare l'operatività. Il consiglio quindi è quello di non abusare di questo strumento (ad esempio trasferendo ad ogni chiamata interi archivi del gestionale), ma sfruttarlo in modo più puntuale.

(vedi esempio Chiamate remote)

CAMPI OBBLIGATORI E GESTIONE DEGLI ERRORI

E' possibile per ogni elemento di tipo input (esclusi i pulsanti) decidere se è obbligatorio inserire un contenuto, bloccare l'esecuzione di azioni in caso di contenuto obbligatorio mancante o per inserimento di dati non validi.

- Il campo è reso obbligatorio se la proprietà _WINOTEMPTY$ = "S": il campo viene evidenziato con un bordo rosso.
- Se ci sono pulsanti che scatenano azioni legate al contenuto dei campi obbligatori (tipicamente sono pulsanti di conferma) è possibile inibire l'evento ON_PRESS del pulsante se nella definizione del pulsante stesso è stata valorizzata la variabile _WICHKEMPTY$ = "S"
- E' possibile definire il numero massimo di caratteri contenuti in una casella di testo valorizzando la variabile _WIMAXCAR = <num_car>
- E' possibile validare il contenuto di una casella di input e segnalare un errore posizionando il focus sulla casella di input stessa mediante l'istruzione
  WCALL "SETERROR",ID_ICOD,"<testo_di_errore>"
- E' comunque possibile forzare il focus su un campo di input usando l'istruzione
  WCALL "FOCUSINPUT",ID_IDATA

(vedi esempio Campi obbligatori)

LAYOUT DELLE LISTE

Per tutti i tipi di lista è possibile definire un layout oltre a quello di default:

1. Numero di elementi per riga: _WLNELRIGA (massimo 2)
2. Dimensione dell'icona: _WLDIMICO (valori da 1 a 10 intesi come moltiplicatori)
3. Posizione dell'icona: _WLPOSICO$ (valori "L","U","R","D": prima lettere della posizione in inglese, es. Left -> "L")

L'utente potrà sempre decidere la visualizzazione a prescindere da come è stata decisa dal programmatore agendo sul menu di Passmobile (icona ingranaggi in alto a destra)

(vedi esempio Layout liste)

LETTURA E SCRITTURA DI PARAMETRI IN LOCALE

Tramite le istruzioni GETLOC e PUTLOC è possibile gestire a livello di dispositivo mobile il salvataggio e la lettura di parametri in locale.

I dati sono gestiti in JSON in modo trasparente e possono essere di 2 tipi:

1. Persistenti ("P"): vengono salvati su file system (e sono condivisibili tra i vari sprix-mobile all'interno della stessa passapp)
2. Temporanei ("T"): vengono salvati solo in memoria RAM

Il programmatore ha a disposizione le seguenti istruzioni per scrivere:

PUTLOC_STR “<tipo>”,”<chiave>”,”<valore>”: scrive in locale un dato di tipo stringa

PUTLOC_NUM “<tipo>”,”<chiave>”,”<valore>”: scrive in locale un dato di tipo numerico

PUTLOC_ARRAY “<tipo>”,”<chiave>”,”<valore>”: scrive in locale un dato di tipo array

Dove <tipo> può essere "P" o "T", <chiave> il nome dell'elemento del JSON e <valore> il valore dell'elemento.

LETTURA E SCRITTURA DI PARAMETRI IN LOCALE

Analogamente per leggere sono disponibili le seguenti istruzioni:

GETLOC_STR “<tipo>”,”<chiave>”,”<valore>”: legge in locale un dato di tipo stringa

GETLOC_NUM “<tipo>”,”<chiave>”,”<valore>”: legge in locale un dato di tipo numerico

GETLOC_ARRAY “<tipo>”,”<chiave>”,”<valore>”: legge in locale un dato di tipo array

Con lo stesso significato dei parametri.

ATTENZIONE: il tipo identifica univocamente il JSON in cui sono stati salvati i parametri, quindi nomi di chiavi uguali ma con tipo differente referenziano oggetti differenti.

Sono disponibili anche le seguenti 2 istruzioni:

1. STORELOC "P": salva forzatamente i dati persistenti nel momento in cui viene invocata e non attende che ci pensi l'esecutore
2. CLEARLOC "<tipo>": cancella tutti i dati del tipo selezionato

Tutte le istruzioni indicate valorizzano le _ERRLOC$ in caso di errore.

(Vedi esempio Gestione parametri locali)

UTILIZZO DEL GPS

Dalla versione 3.4 di Passmobile è possibile accedere al GPS interno dei dispositivi mobile per raccogliere le informazioni sulla posizione del dispositivo.

Sono disponibili 2 modalità di interazione con il GPS:

1. SENDGPSDATA: i dati del GPS vengono inviati in tempo reale al server
2. GETGPSDATA: i dati del GPS vengono letti solo localmente

Le 2 funzionalità sono accessibili tramite le istruzioni di servizio WCALL:

1. WCALL "SENDGPSDATA",n: dove n è la frequenza di lettura in secondi
2. WCALL "GETGPSDATA",0

UTILIZZO DEL GPS - SENDGPSDATA

Nel caso di invio dei dati tramite SENDGPSDATA, i dati verranno inviati ogni n secondi al server, oppure solo quando viene invocata se n=0. Se n=-1 si interrompe una eventuale invio temporizzato avviato in precedenza.

I dati potranno essere letti lato server (array di dati) con il seguente significato:

"num. dati nell'array" :	_GPSNUM
"datainvio" :	_GPSDATA$(CONT)
"orainvio" :	_GPSORA$(CONT)
"siglaazienda" :	_GPSAZIENDA$(CONT)
"utente" :	_GPSUTENTE$(CONT)
"iddisp" :	_GPSIDDISP$(CONT)
"nomedis" :	_GPSNMDISP$(CONT)
"dispositivo"	_GPSDISP$(CONT)

"latitudine" :	_GPSLAT$(CONT)
"longitudine" :	_GPSLONG$(CONT)
"indirizzo" :	_GPSIND$(CONT)
"localita" :	_GPSLOC$(CONT)
"idutente" :	_GPSIDU$(CONT)
"nomeapp":	_GPSAPP$(CONT)
"idapp":	_GPSIDAPP$(CONT)

UTILIZZO DEL GPS - SENDGPSDATA

Lato server i dati possono essere letti da Sprix tramite l'istruzione

GETGPSDATA ANNOMESE,GIORNO,AZIENDA,IDUTENTE,IDAPP

in cui solo il parametro ANNOMESE è obbligatorio (Es. GETGPSDATA "202001","","","","")

ATTENZIONE: a seconda del dispositivo utilizzato può capitare che, per le policy di risparmio energetico, quando l'app è in background o il display spento, i dati gps vengono resi disponibili solo quando cambia la posizione anche se è stato impostato un invio temporizzato

UTILIZZO DEL GPS – GETGPSDATA (lato mobile)

Nel caso di lettura dei dati tramite GETGPSDATA, n è sempre uguale a 0.

I dati saranno disponibili solo gestendo l'evento ON_GETGPSDATA in cui sarà possibile leggere le seguenti variabili (gli stessi array ma di un solo elemento):

- "datainvio" : _GPSDATA$(1)
- "orainvio" : _GPSORA$(1)
- "siglaazienda" : _GPSAZIENDA$(1)
- "utente" : _GPSUTENTE$(1)
- "latitudine" : _GPSLAT$(1)
- "longitudine" : _GPSLONG$(1)
- "indirizzo" : _GPSIND$(1)
- "localita" : _GPSLOC$(1)
- "idutente" : _GPSIDU$(1)
- "nomeapp": _GPSAPP$(1)
- "idapp": _GPSIDAPP$(1)

UTILIZZO DEL GPS - GETGPSDATA

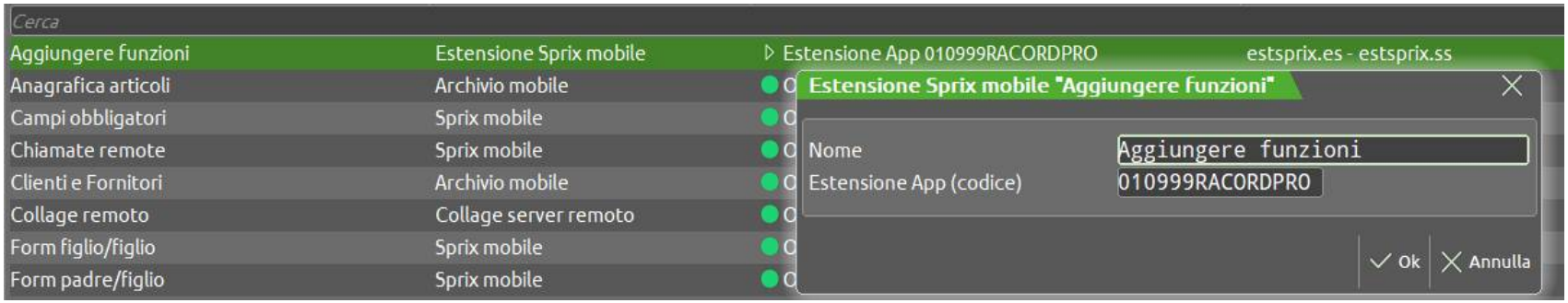
L’istruzione GETGPSDATA interrompe eventuali invii temporizzati che dovranno quindi essere rispristinati dentro l’etichetta di callback rifacendo una nuova WCALL “SENDGPSDATA”,n.

ATTENZIONE: è consigliato evitare di vincolare il flusso del programma alla ricezione dei dati GPS in quanto, a seconda del dispositivo utilizzato, ci possono volere anche diversi secondi. Se l’aggiornamento dei dati GPS viene in qualche modo vincolato all’operatività utente potrebbero verificarsi rallentamenti indesiderati.

(Vedi esempi Gestione del GPS)

ESTENSIONE SPRIX-MOBILE

Utilizzando l'elemento Estensione Sprix Mobile è possibile aggiungere funzionalità ad applicazioni fatte da terzi di cui non si possiede il codice. Per estendere una app è sufficiente indicare il suo codice nel menu di configurazione dell'elemento Estensione Sprix Mobile



E' poi sufficiente aggiungere codice come si farebbe in un normale Sprix Mobile.
L'elemento Estensione Sprix Mobile comparirà nella app come una nuova voce di menu.

ESTENSIONE COLLAGE-MOBILE

Analogamente alle Estensioni Sprix Mobile, nella configurazione dell'elemento deve essere specificato il codice della app che si vuol estendere.

Nelle estensioni collage mobile possono essere utilizzate solo le seguenti etichette

ON_IN_<"codice_oggetto">

ON_OUT_<"codice_oggetto">

ON_SHOW_<"codice_oggetto">

N.B.: Il codice oggetto viene recuperato abilitando la modalità sviluppo dalle impostazioni di Passmobile e poi facendo un long press sull'elemento desiderato. L'id oggetto si ottiene utilizzando l'istruzione WGETOID

Nell'evento ON_SHOW è possibile definire l'inserimento di nuovi pulsanti mediante il solito costrutto.

Solo in questo contesto sono disponibili 2 ulteriori variabili di struttura per gli input:

- _WIINSRIFID: id dell'oggetto di riferimento prima o dopo del quale inserire l'oggetto
- _WIINSPOS$: posizione (“AFTER”,“BEFORE”,”BEFOREFIRST,”AFTERLAST”)
    - AFTER: dopo l'elemento specificato da _WIINSRIFID
    - BEFORE: prima dell'elemento specificato da _WIINSRIFID
    - BEFOREFIRST: prima del primo oggetto che si trova nel form
    - AFTERLAST: dopo l'ultimo oggeto che si trova nel form

ESTENSIONE COLLAGE-MOBILE

Sui nuovi pulsanti inseriti possono poi essere gestiti opportuni eventi ON_PRESS

Nelle etichette di gestione degli eventi ON_PRESS è tutto ammesso tranne:

- fare altre estensioni collage: ad esempio fare un form e poi un'altra estensione collage che aggiunge un pulsante in quel form
- fare WSHOWFORM su form diversi da quello di partenza: si possono visualizzare tutti i form nuovi creati nella ON_SHOW ma dei form dell'app "originale" si può fare la WSHOWFORM solo del form in cui è stato inserito il nuovo pulsante

(vedi esempi Estensione Collage Mobile e Aggiungere funzioni)

PASSEPARTOUT®
Software e servizi gestionali

Corso Sprix Mobile

Gianluca Suzzi