BADA

Quick Guide

Jesús Martín García (47869183L)
Maig de 2011

Continguts

Idees generals

La versió utilitzada ha sigut BADA 1.2, que es pot aconseguir juntament amb el SDK només per a Windows. El fet que no estigui per a linux és un inconvenient important a l'hora de fer debug de les connexions.

El SDK té incorporat una sèrie de programes base per a diferents aplicacions que es poden aprofitar combinant-los adequadament. Aquests exemples són vitals ja que de molts temes no hi ha gens documentació.

Per altra banda l'estil de programació es basa en espais de noms. Existeix una gran diversitat d'espais de noms, cadascun adreçat a resoldre un problema en concret: connexió, tractament de dades, gps, imatge, so,... Quan necessitem una funció en concret només hem de buscar el seu namespace i veure la documentació.

Les idees generals a tenir en compte en la filosofía bada són:

- Qui ho crea ho destrueix (objectes i vectors).
- Amplia modularitat sense crear dependències innecessàries.
- Alta capacitat de càlcul aprofitable mitjançant threads.
- Les limitacions ja es solucionaràn a la següent versió.

L'últim punt fa referència a l'inminent sortida de BADA 2.0, que s'ha convertit en l'excusa per a no arreglar els bugs coneguts del sdk.

User Interface

La UI és similar a la resta de plataformes: contenidors (formularis), elements d'interacció (llistes, botons, etiquetes,...) i listeners per als events.

Els formularis es poden crear o bé mitjançant ajuda gràfica o bé directament sobre codi. Jo he utilitzat les dues maneres tot i que acaben sent el mateix.

Forma gràfica:

1) Crearem la plantilla en xml (Resources -> Form -> Open Insert Wizard).
2) Crearem la classe associada al xml (clic dret -> Add Class).
3) Recuperarem tots els controls del xml al arxiu de la classe a base de crear punters per a cada element. Això generalment es fa a la funció d'inicialització (que no és la constructora).

Directament amb codi:

1) Crearem les classes en c++ heretant de la classe Osp::Base::Form,
2) A la constructora utilitzarem la classe estàtica Osp::Base::Form::Construct,

passant-li els paràmetres adicionals (softkeys, mida, color, ...)
3) Creem directament els controls indicant per a cada un la posició (setPosition), color, text,...

Una vegada ja tenim els formularis creats ens interessarà poder canviar d'un a l'altre interactivament. Això ho farem afegint el formulari al frame actual i refrescant la pantalla:

1) Obtenir punter a l'aplicació.
2) Obtenir el frame de l'aplicació.
3) Afegir al frame un nou formulari (crear instància) i indicar que és el default.
4) Inicialitzar el formulari.
5) Refrescar (RequestRedraw).

Per a poder fer els primers dos punts hem d'incloure les llibreries necessàries de sistema i aplicació.

Els controls i listeners funcionan de la mateixa manera que en qualsevol altre sistema basat en events: els controls creen events (click, changue, hover,...) i els listeners s'encarreguen de capturar aquests events. Tot es pot fer mitjançant clic dret -> add event handler sobre el control que volguem.

Comunicació http

Per a poder implementar la comunicació http / http hem d'utilitzar el programa d'exemple HttpClient. Aquest ens dóna tota l'estructura necessària per a iniciar una nova comunicació i rebre el resultat.

L'estructura del programa d'exemple consta d'una funció principal i tota la sèrie de listeners. Faig una breu descripció de cadascún:

- Funció principal: dóna valor als paràmetres (url, uri), genera el header (GET/POST) i el body del missatge. A més és l'encarregat d'obrir una nova transacció i començar el enviament de dades (submit transaction).
- Listener ReadyToRead: recull el resultat del buffer de missatges. Pot ser cridada més d'una vegada si el missatge és molt llarg.
- Listener ReadyToWrite: utilitzat en comunicacions multi-part, escriu el nou body i l'envia.
- Listener OnTransactionHeaderComplete: salta al acabar d'enviar el header i és on s'ha d'incorporar la autenticació (user/pass).
- Listener OnCertReceived: salta si existeix un certificat i retorna el seu text.
- Listeners OnTransactionComplete / OnTransactionError: s'executen quan s'ha

acabat la transacció, en el cas de complete és si ha estat exitosa i crida al ReadyToRead.

Amb això ja podem fer una primera connexió, però el major problema és que es tracta d'una comunicació asíncrona. Per tant al acabar la primera funció ens retorna el control i ens hem de buscar la vida per a implementar el sincronisme. La solució passa per utilitzar threads.

Event-Driven-Thread és la classe de thread que utilitzarem ja que és la única capaç de soportar les funcions asíncrones. No hi ha cap implementació al sdk ni a la documentació però sí un exemple al fòrum aprovat per badateam i que tothom està utilitzant als seus programes.

1) Crearem thread amb el contingut del HttpClient. La diferència es que li passarem un listener per paràmetre al constructor.
2) Crearem un handler. Aquest és l'encarregat de 'penjar-se' fins que el HttpClient no hagi acabat la connexió. El funcionament és basa en l'ús d'un monitor (semàfor amb paràmetre = 0). Ha d'implementar tots els listeners del HttpClient.
3) Quan el handler crea un nou objecte HttpClient es passa a si mateix com a paràmetre listener, de manera que el handler passa a ser el listener del HttpClient. Ho representarem amb una variable del tipus listener dins el HttpClient que agafarà com a valor el paràmetre passat.
4) Dins les funcions listener del HttpClient farem una redirecció al seu listener (handler). Simplement farem un listener-> OnTransactionComplete dins el OnTransactionComplete del HttpClient.
5) Dins la implementació dels listeners del handler farem que pari el monitor amb l'ordre monitor->notifyAll(). Així el handler deixarà d'esperar i s'acaba el sincronisme.

Com que la implementació de WireShark per a Windows no permet escanejar la interfície loopback, hem de crear un servidor a part per a poder fer les proves.

Mapes Decarta

Bada no té suport per a Google Maps, per contra té un acord amb Decarta i integra els seus mapes com a part del sdk. Abans d'utilitzar aquest servei hem d'anar a la web oficial i registrar-nos per obtenir un número d'usuari.

El seu funcionament és similar a Gmaps, utilitzant markers (punts senyalats), globus de text i tantes figures geomètriques volguem. Tot i així es troba molt més limitat, per exemple només podem posar un globus de text en pantalla a la vegada.

Instal·lacio al dispositiu

Una vegada hem acabat el nostre software l'hem de pujar al mòbil. Aquesta operació és relativament senzilla.

Per començar ens hem de registrar a la pàgina de desenvolupament i registrar la nostra aplicació. Hem de fer un arxiu manifest.xml on hi han detallades les funcionalitats que utilitzarem (gps, camera, arxius,...) i aquest arxiu ha de ser incorporat a la nostra carpeta de treball per a poder pujar l'aplicació.

Aquest arxiu es genera corrupte. Per a poder instalar l'aplicació al mòbil has d'editar l'arxiu i copiar els valors de les rutes de les imatge Splash.png al camp buit del xml (Ticker). Una solució alternativa és esborrar el primer privilegi, però llavors l'arxiu no és vàlid.

Una vegada feta la modificació es compila utilitzant la opció Target. Connectem el mòbil en depuració usb i cliquem en Run-Target-Application. El software es copiarà al telèfon fins que l'esborrem com una aplicació més.

Bugs

Durant el desenvolupament del projecte he trobat tres bugs importants que espero arreglin a la pròxima revisió i que per ara no s'hi pot fer res:

- ArrayList<Object>: s'han detectat problemes i al fòrum no han sapigut donar una resposta exacte. A vegades no recupera bé els elements i produeix un segmentation fault que penja el programa.
- Multi-part: al crear missatges multi-part en http el sdk genera missatges que no són reconeguts per cap aplicació. Utilitzat un analitzador de tràfic es pot veure com relitza l'enviament de totes les parts (en el meu cas una foto), però el destí no reconeix que es tracta d'un multipart tot i haver-ho creat mitjançat un missatge hardcoded. No hi ha cap solució compartida al fòrum ni cap testimoni de ningú que ho hagi aconseguit fer anar.

Enllaços

http://developer.bada.com
Web principal per a desenvolupadors de bada. Aquí està el fòrum on es poden trobar la majoria de problemes i solucions. Recentment han afegit una secció de tips per als desenvolupadors.

http://www.badadev.com/
Aquesta web no forma part de l'equip de bada però el fòrum general hi referencia moltes solucions. Es tracta d'una web de tutorials, problemes i solucions.

http://www.badaspain.com/
Web espanyola de bada, no aporta gran cosa al development però de tant en tant pots trobar enllaços interesants a blogs on hi han solucions.