Πτυχιακή Εργασία

Τίτλος: Ανάπτυξη mobile εφαρμογών και services στην πλατφόρμα Android

Μουτζουρίδης Γεώργιος (ΑΜ: 001462)

Επιβλέπων καθηγητής: Σφέτσος Παναγιώτης

Περίληψη

Στόχος της παρούσας πτυχιακής εργασίας είναι η απόκτηση των γνώσεων που απαιτούνται για την ανάπτυξη εφαρμογών για το λειτουργικό Android. Η εργασία περιλαμβάνει θεωρητικό και πρακτικό μέρος. Για το πρακτικό μέρος αναπτύχθηκε μία εφαρμογή με την οποία ο χρήστης μπορεί να δει το πρόγραμμα των προβολών για τις ταινίες που παίζονται σε διάφορες πόλεις της χώρας, όπως επίσης και να πάρει κάποιες πληροφορίες για αυτές. Στο θεωρητικό μέρος, στην πρώτη ενότητα γίνεται μία γενική περιγραφή των χαρακτηριστικών του Android. Στη δεύτερη ενότητα γίνεται αναφορά στα βήματα που απαιτούνται για τη κατάλληλη διαμόρφωση του περιβάλλοντος. Στη τρίτη ενότητα γίνεται λόγος για την διεπαφή χρήστη, ενώ η τέταρτη και η πέμπτη αναφέρονται στη χρήση γραφικών και animations στο Android. Στην έκτη ενότητα περιγράφεται η ασφάλεια στο Android. Ακολουθεί μία ενότητα για τις δοκιμές στο Android, ενώ η τελευταία αναφέρεται στα Google Services. Τέλος, υπάρχει ειδικό παράρτημα όπου περιγράφεται αναλυτικά η εφαρμογή που έχει αναπτυχθεί.

Περιεχόμενα

Γενικά για το Android

Τι είναι το Android?

Το Android είναι ένα λειτουργικό σύστημα βασισμένο στον πυρήνα του Linux, για φορητές συσκευές όπως είναι τα smartphones και οι υπολογιστές τύπου tablet. Αναπτύσσεται από το Open Handset Alliance του οποίου επικεφαλής είναι η Google.

Το Android προσφέρει μία ενοποιημένη προσέγγιση για την ανάπτυξη εφαρμογών για φορητές συσκευές. Οι προγραμματιστές μπορούν να αναπτύσσουν εφαρμογές για το Android, οι οποίες μπορούν να τρέξουν σε διαφορετικές συσκευές με λειτουργικό Android.

Η Google διέθεσε την πρώτη beta έκδοση του Android Software Development Kit (SDK) το 2007, ενώ η πρώτη εμπορική έκδοση, η Android 1.0, κυκλοφόρησε το Σεπτέμβριο του 2008. Η τελευταία έκδοση του Android SDK είναι η 5.1 Lollipop. Κυκλοφόρησε το Μάρτιο του 2015.

Ο πηγαίος κώδικας του Android διατίθεται σε άδειες ελεύθερου και ανοιχτού λογισμικού. Το μεγαλύτερο μέρος του κώδικα διατίθεται από την Google με την Apache Licence 2.0, ενώ το υπόλοιπο μέρος, που αφορά τις αλλαγές στον πυρήνα του Android, διατίθεται με την GNU General Public License 2.

Τα χαρακτηριστικά του Android

Το Android είναι ένα λειτουργικό σύστημα με πλούσιες δυνατότητες, ορισμένες από τις οποίες αναφέρονται παρακάτω:

Χαρακτηριστικό	**Περιγραφή**
Καλοσχεδιασμένο περιβάλλον εργασίας χρήστη	Η βασική οθόνη του Android παρέχει ένα όμορφο και διαισθητικό περιβάλλον εργασίας χρήστη.
Συνδεσιμότητα	GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC and WiMAX.
Αποθήκευση δεδομένων	Η SQLite, μία ελαφριά σχεσιακή βάση δεδομένων, μπορεί να χρησιμοποιηθεί για την αποθήκευση δεδομένων.
Υποστηριζόμενοι τύποι πολυμέσων	H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, and BMP
Αποστολή μηνυμάτων	SMS και MMS

Πρόγραμμα περιήγησης Web	Βασίζεται στην WebKit, μηχανή διατάξεων (layout engine) ανοιχτού λογισμικού, και στην μηχανή JavaScript του Chrome V8. Υποστηρίζει HTML5 και CSS3.
Τεχνολογία πολλών σημείων επαφής	Το Android υποστηρίζει εγγενώς την τεχνολογία πολλών σημείων επαφής, η οποία αρχικά ήταν διαθέσιμη σε κινητά όπως το HTC Hero.
Πολυδιεργασία	Οι χρήστες μπορούν να μεταβούν από την μία εφαρμογή στην άλλη. Επιπλέον, διαφορετικές εφαρμογές μπορούν να τρέχουν ταυτόχρονα.
Υποστήριξη γλωσσών	Το Android υποστηρίζει πολλές γλώσσες.
GCM	Το Google Cloud Messaging είναι μία υπηρεσία η οποία απλοποιεί την αποστολή σύντομων μηνυμάτων σε συσκευές Android, από έναν διακομιστή.
Wi-Fi Direct	Τεχνολογία η οποία επιτρέπει στις εφαρμογές να εντοπίσουν η μία την άλλη και να συνδεθούν απευθείας.
Android Beam	Δημοφιλής τεχνολογία, βασισμένη στο NFC, η οποία επιτρέπει στους χρήστες να ανταλλάσσουν πολύ εύκολα δεδομένα, αγγίζοντας απλώς δύο κινητά ταυτόχρονα.

Οι εφαρμογές του Android

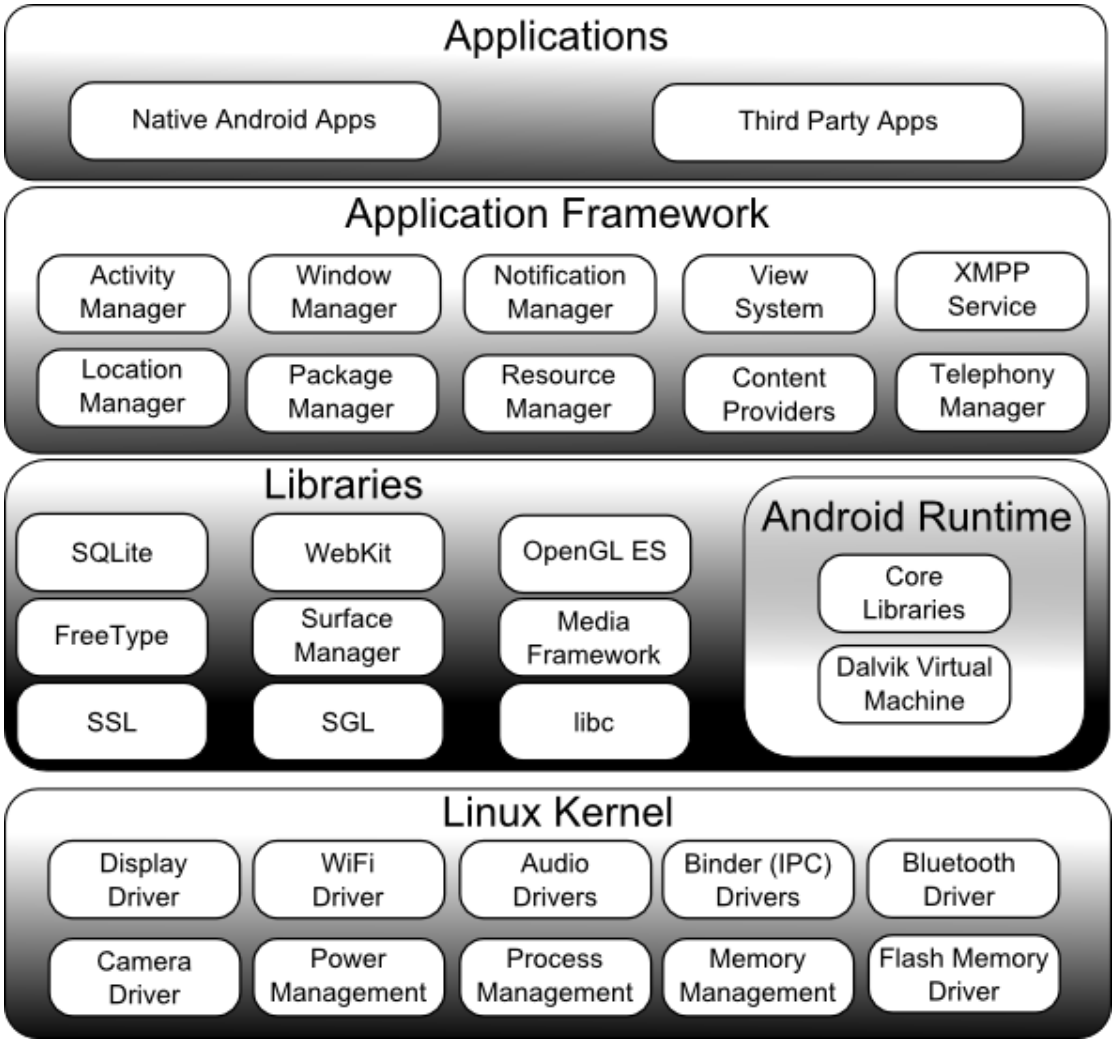
Οι εφαρμογές για το Android αναπτύσσονται συνήθως σε γλώσσα Java με χρήση του Android Software Development Kit. Μπορεί επίσης να γίνει ανάπτυξη εφαρμογών με χρήση γλωσσών όπως η C και η C++, οι οποίες παράγουν εγγενή κώδικα, με χρήση του Android Native Development Kit.

Αφού ολοκληρωθούν, οι εφαρμογές μπορούν να πουληθούν εύκολα μέσω διαδικτυακών καταστημάτων όπως είναι το Google Play ή το Amazon Appstore. Οι εφαρμογές εγκαθίστανται από ένα αρχείο το οποίο έχει κατάληξη .apk (android package).

Το Android είναι εγκατεστημένο σε εκατοντάδες εκατομμύρια φορητές συσκευές, σε περισσότερες από 190 χώρες στον κόσμο. Είναι η πιο διαδεδομένη πλατφόρμα για φορητές συσκευές, ενώ κάθε μέρα ενεργοποιούνται περισσότερες από 1 εκατομμύριο συσκευές Android σε όλο τον κόσμο.

Η αρχιτεκτονική του Android

Το Android αποτελείται από μία στοίβα στοιχείων λογισμικού, η οποία διαιρείται σε 5 τμήματα και 4 επίπεδα όπως φαίνεται στο διάγραμμα που ακολουθεί.



Πυρήνας Linux

Στο κατώτερο επίπεδο, το οποίο επικοινωνεί με το υλικό, βρίσκεται μία τροποποιημένη έκδοση του πυρήνα Linux. Από τον Απρίλιο του 2014, οι συσκευές Android χρησιμοποιούν κυρίως τις εκδόσεις 3.4 ή 3.10 του πυρήνα Linux. Ο πυρήνας παρέχει βασικές λειτουργίες του συστήματος, όπως είναι η διαχείριση της δικτύωσης, των διεργασιών, της μνήμης και συσκευών όπως η κάμερα, το πληκτρολόγιο και η οθόνη. Επιπλέον, διαθέτει μία μεγάλη ποικιλία προγραμμάτων οδήγησης.

Βιβλιοθήκες

Πάνω από τον πυρήνα υπάρχει μια σειρά βιβλιοθηκών γραμμένες σε εγγενή κώδικα C ή C++, στις οποίες περιλαμβάνεται η μηχανή προγράμματος περιήγησης Web WebKit, η γνωστή βιβλιοθήκη libc, η βάση δεδομένων SQLite, η οποία είναι χρήσιμη για την αποθήκευση δεδομένων και την κοινή χρήση τους με άλλες εφαρμογές, βιβλιοθήκες για την αναπαραγωγή ήχου και βίντεο, βιβλιοθήκες SSL υπεύθυνες για την ασφαλή μετάδοση δεδομένων στο Internet κλπ.

Το περιβάλλον εκτέλεσης (runtime) του Android

Το περιβάλλον εκτέλεσης βρίσκεται και αυτό στο δεύτερο επίπεδο μετρώντας από κάτω προς τα πάνω. Μέχρι την έκδοση 4.4 του Android χρησιμοποιούνταν η Dalvik Virtual Machine, ενώ από την 5η έκδοση και έπειτα η Dalvik Virtual Machine έχει αντικατασταθεί από την εικονική μηχανή Android Runtime (ART). Και οι δύο εικονικές μηχανές είναι ειδικά σχεδιασμένες και βελτιστοποιημένες για το Android.

Η εικονική μηχανή Dalvik επιτρέπει σε κάθε εφαρμογή να εκτελείται στην δική της διεργασία, με το δικό της στιγμιότυπο (instance) της Dalvik. Σε αντίθεση με την Dalvik η οποία χρησιμοποιεί just-in-time μεταγλώττιση, η ART χρησιμοποιεί ahead-of-time μεταγλώττιση κατά την εγκατάσταση της εφαρμογής στην συσκευή. Έτσι επιτυγχάνεται καλύτερη απόδοση εφόσον δεν χάνεται χρόνος για να γίνει η μεταγλώττιση του κώδικα όταν τρέχει η εφαρμογή. Επιπλέον, η ART χρησιμοποιεί βελτιωμένο συλλέκτη σκουπιδιών (garbage collector). Στα μειονεκτήματα της ART είναι ότι απαιτείται περισσότερος χρόνος για να γίνει η εγκατάσταση, και ότι ο κώδικας μηχανής που προκύπτει από την μεταγλώττιση είναι μεγαλύτερος σε μέγεθος οπότε απαιτείται περισσότερος χώρος για την αποθήκευσή του.

Το περιβάλλον εκτέλεσης παρέχει επίσης μία σειρά από βιβλιοθήκες οι οποίες επιτρέπουν στους προγραμματιστές να γράφουν εφαρμογές χρησιμοποιώντας πολλές από τις κλάσεις οι οποίες υπάρχουν στην Java Standard Edition.

Το framework για τις εφαρμογές

Το επίπεδο του framework για τις εφαρμογές παρέχει σε αυτές πολλές υπηρεσίες υψηλού επιπέδου με τη μορφή βιβλιοθηκών Java, οι οποίες έχουν σχεδιαστεί ειδικά για το Android. Οι προγραμματιστές μπορούν να χρησιμοποιήσουν αυτές τις υπηρεσίες στις εφαρμογές τους.

Εφαρμογές

Οι εφαρμογές βρίσκονται στο ανώτερο επίπεδο. Παραδείγματα αποτελούν οι εφαρμογές Contacts, Browser, Calendar, Calculator, Games κλπ.

Στοιχεία λογισμικού (components) εφαρμογών

Τα στοιχεία λογισμικού είναι τα δομικά στοιχεία μιας εφαρμογής για Android. Το Android επιτρέπει την χρήση των στοιχείων λογισμικού μιας εφαρμογής από άλλες εφαρμογές.

Τέσσερα είναι τα κύρια στοιχεία λογισμικού που μπορούν να χρησιμοποιηθούν σε μία εφαρμογή Android:

Στοιχείο λογισμικού	Περιγραφή
Δραστηριότητες (Activities)	Μία Δραστηριότητα είναι, γενικά, ο κώδικας για μία εργασία, επικεντρωμένη στο χρήστη. Συνήθως περιλαμβάνει και μία διεπαφή χρήστη (user interface). Κάποια από τις δραστηριότητες της εφαρμογής αποτελεί το σημείο εισόδου της εφαρμογής. Μία εφαρμογή μπορεί να έχει πάνω από ένα σημεία εισόδου.
Υπηρεσίες (Services)	Μία υπηρεσία είναι ένα κομμάτι κώδικα το οποίο τρέχει στο παρασκήνιο για σχετικά μεγάλο χρονικό διάστημα. Επιπλέον, δεν περιλαμβάνει κάποια διεπαφή χρήστη.
Δέκτες Μεταδόσεων (Broadcast Receivers)	Ο Δέκτης Μετάδοσης είναι ένα στοιχείο λογισμικού το οποίο απαντάει σε μεταδόσεις για ολόκληρο το σύστημα (system-wide broadcasts), οι οποίες δημιουργούνται είτε από το λειτουργικό σύστημα είτε από κάποια άλλη εφαρμογή. Για παράδειγμα, μια μετάδοση η οποία ανακοινώνει ότι η οθόνη έχει σβήσει ή ότι η μπαταρία είναι χαμηλή.
Πάροχοι Περιεχομένου (Content Providers)	Ένας Πάροχος Περιεχομένου διαχειρίζεται τα κοινόχρηστα δεδομένα μίας εφαρμογής. Τα δεδομένα μπορούν να αποθηκευτούν στο σύστημα αρχείων, σε μία βάση SQLite, στο διαδίκτυο, ή σε κάποια άλλη τοποθεσία μόνιμης αποθήκευσης στην οποία έχει πρόσβαση η εφαρμογή. Μέσω του Παρόχου Περιεχομένου, άλλες εφαρμογές μπορούν να έχουν πρόσβαση στα δεδομένα και, αν το επιτρέπει ο Πάροχος Περιεχομένου, να τα τροποποιήσουν.

Άλλα στοιχεία μιας εφαρμογής Android

Υπάρχουν άλλα στοιχεία τα οποία χρησιμοποιούνται για την κατασκευή των προαναφερθέντων στοιχείων, την μεταξύ τους επικοινωνία και τη σύνδεσή τους.

Στοιχεία	Περιγραφή
Τμήματα (Fragments)	Ένα τμήμα αποτελεί μέρος της συμπεριφοράς ή της διεπαφής χρήστη μιας Δραστηριότητας. Πρέπει πάντα να είναι ενσωματωμένο σε μία Δραστηριότητα, ενώ μπορεί να χρησιμοποιείται από πολλές Δραστηριότητες.
Views	Στοιχεία μιας διεπαφής χρήστη, όπως κουμπιά, αναπτυσσόμενες λίστες, πλαίσια κειμένου, κουμπιά επιλογής (radio buttons), πλαίσια ελέγχου (text boxes) κλπ.
Διατάξεις (Layouts)	Οι διατάξεις καθορίζουν την οπτική δομή μιας διεπαφής χρήστη, όπως είναι η διεπαφή χρήστη μιας δραστηριότητας.

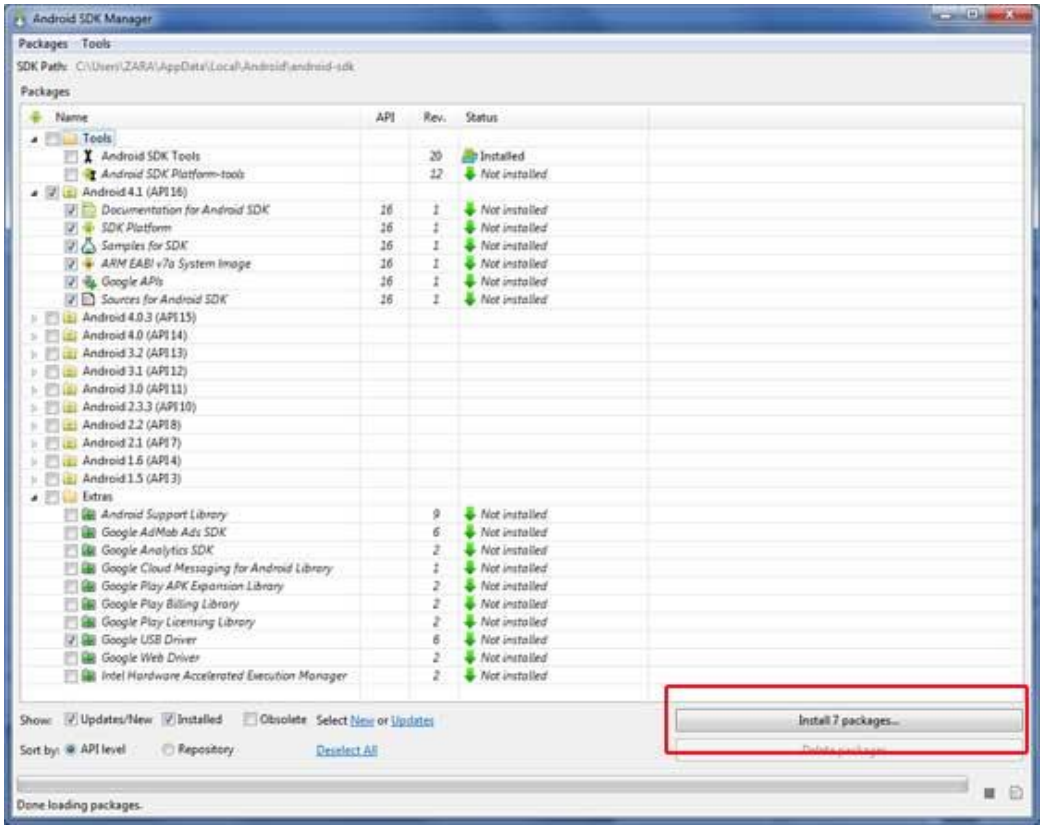
Προθέσεις (Intents)	Μηνύματα τα οποία χρησιμοποιούνται για την επικοινωνία διαφορετικών στοιχείων λογισμικού, τα οποία μπορεί να ανήκουν στην ίδια ή και σε διαφορετικές εφαρμογές.
Πόροι (Resources)	Μία εφαρμογή μπορεί να περιλαμβάνει μία σειρά από εικόνες και αρχεία ήχου ή βίντεο, διάφορα XML αρχεία τα οποία περιγράφουν τις διατάξεις, πακέτα γλωσσών κλπ. Όλα αυτά αποτελούν τους πόρους της εφαρμογής και είναι αποθηκευμένα στο αρχείο resources.ap_ εντός του .apk, όπως επίσης και σε υποφακέλους όπως ο drawable για τις εικόνες.
AndroidManifest	Το αρχείο AndroidManifest.xml περιέχει πληροφορίες τις οποίες πρέπει να γνωρίζει το σύστημα προκειμένου να τρέξει μία εφαρμογή. Μεταξύ άλλων στο AndroidManifest περιλαμβάνονται τα εξής: το όνομα του πακέτου Java το οποίο αποτελεί μοναδικό αναγνωριστικό για την κάθε εφαρμογή, η ελάχιστη έκδοση API για την οποία έχει σχεδιαστεί η εφαρμογή, τα στοιχεία λογισμικού της εφαρμογής με τις δυνατότητές τους, τα δικαιώματα που θα πρέπει να έχει η εφαρμογή προκειμένου να έχει πρόσβαση σε προστατευμένα τμήματα του API και σε στοιχεία λογισμικού άλλων εφαρμογών, τα δικαιώματα που πρέπει να έχουν άλλες εφαρμογές προκειμένου να συνδεθούν με τα στοιχεία λογισμικού της εφαρμογής.

Διαμόρφωση Περιβάλλοντος (Setup)

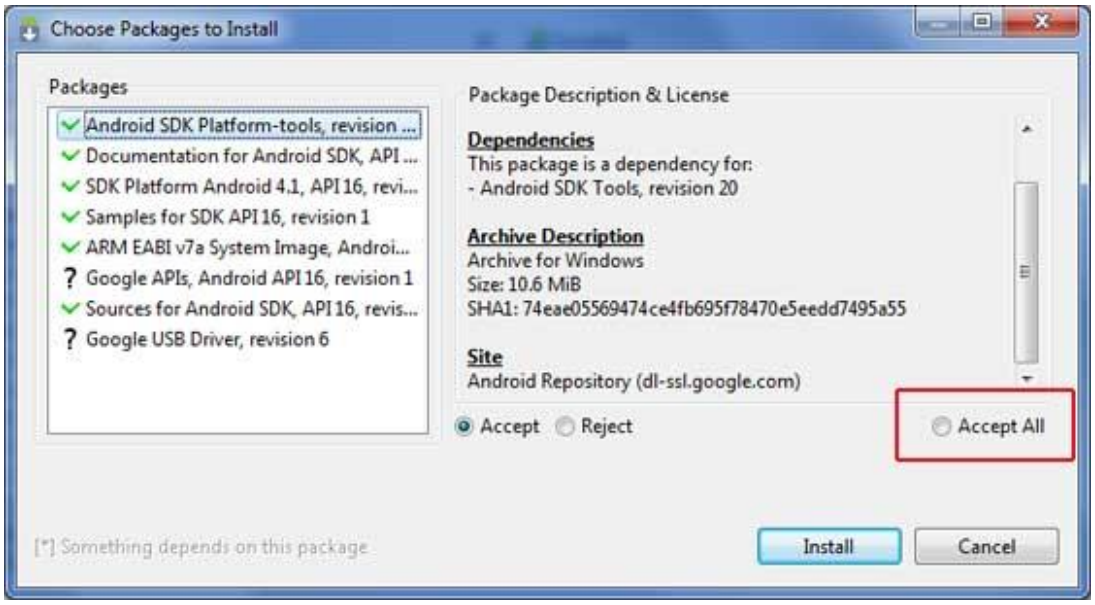
Το Eclipse ήταν παλιότερα το επίσημο IDE (ενσωματωμένο περιβάλλον ανάπτυξης) για την ανάπτυξη εφαρμογών για Android. Πλέον το επίσημο IDE είναι το Android Studio. Ακολουθεί μία αναφορά στα βήματα που απαιτούνται προκειμένου να μπορεί να εργαστεί ένας προγραμματιστής με αυτά τα δύο IDE.

Android SDK

Αρχικά πρέπει να γίνει εγκατάσταση του Android SDK. Αν εκκινήσουμε τον Android SDK Manager θα δούμε το ακόλουθο παράθυρο:



Υπάρχει περίπτωση να πρέπει να εγκατασταθούν κάποια νεότερα πακέτα προκειμένου να ενημερωθεί το Android SDK.



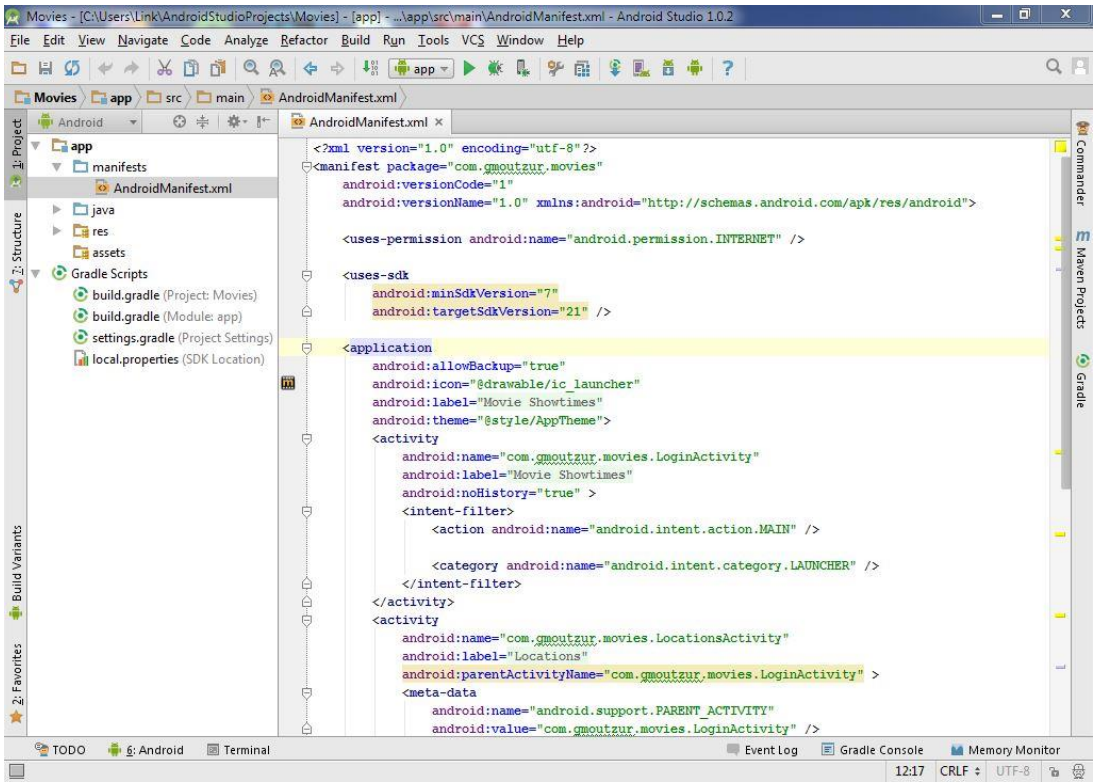
Java Development Kit (JDK)

Για την ανάπτυξη εφαρμογών γραμμένων σε Java απαιτείται επιπλέον η εγκατάσταση του JDK από τον σχετικό ιστότοπο της Oracle. Για το Eclipse προτείνεται η έκδοση του JDK να είναι η 6η ή κάποια νεότερη, ενώ για το Android Studio η 7η ή κάποια νεότερη. Σε ότι αφορά το Eclipse, θα πρέπει να

εγκαταστήσουμε την 32-bit έκδοση αν η έκδοση της Java Virtual Machine είναι 32-bit, ενώ απαιτείται η 64-bit έκδοση του Eclipse αν χρησιμοποιούμε την 64-bit έκδοση της JVM.

Android Studio

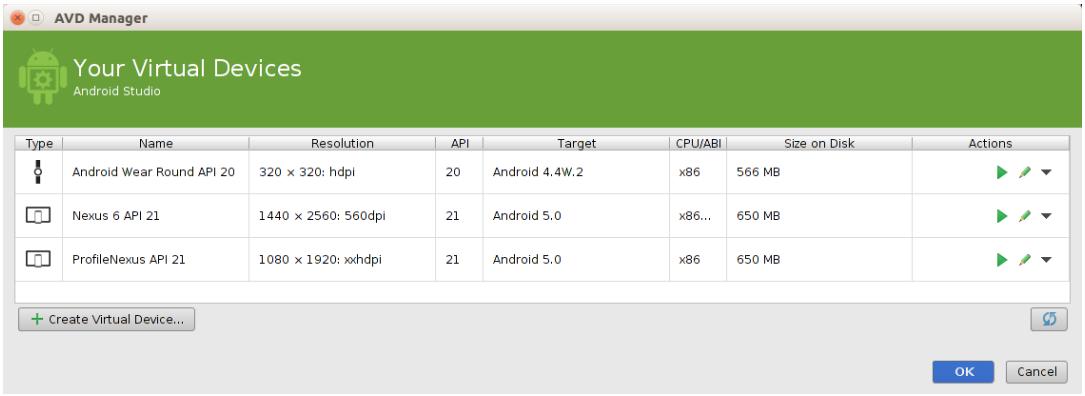
Το Android Studio διατίθεται σε τρεις διαφορετικές εκδόσεις, για Windows, Linux, και Mac OS X.



Ανεξάρτητα από το αν χρησιμοποιούμε το Android Studio ή τη γραμμή εντολών, για να τρέξουμε μία εφαρμογή στον εξομοιωτή θα πρέπει να δημιουργήσουμε μία εικονική συσκευή Android (AVD). Μία AVD έχει τις ρυθμίσεις οι οποίες απαιτούνται προκειμένου ο εξομοιωτής Android να μπορεί να εξομοιώσει μία συγκεκριμένη συσκευή.

Δημιουργία μιας AVD

1. Εκκινούμε τον Διαχειριστή AVD (AVD Manager):
    - Στο Android Studio, επιλέγουμε **Tools > Android > AVD Manager**, ή κάνουμε κλικ στο σχετικό εικονίδιο  στη γραμμή εργαλείων.

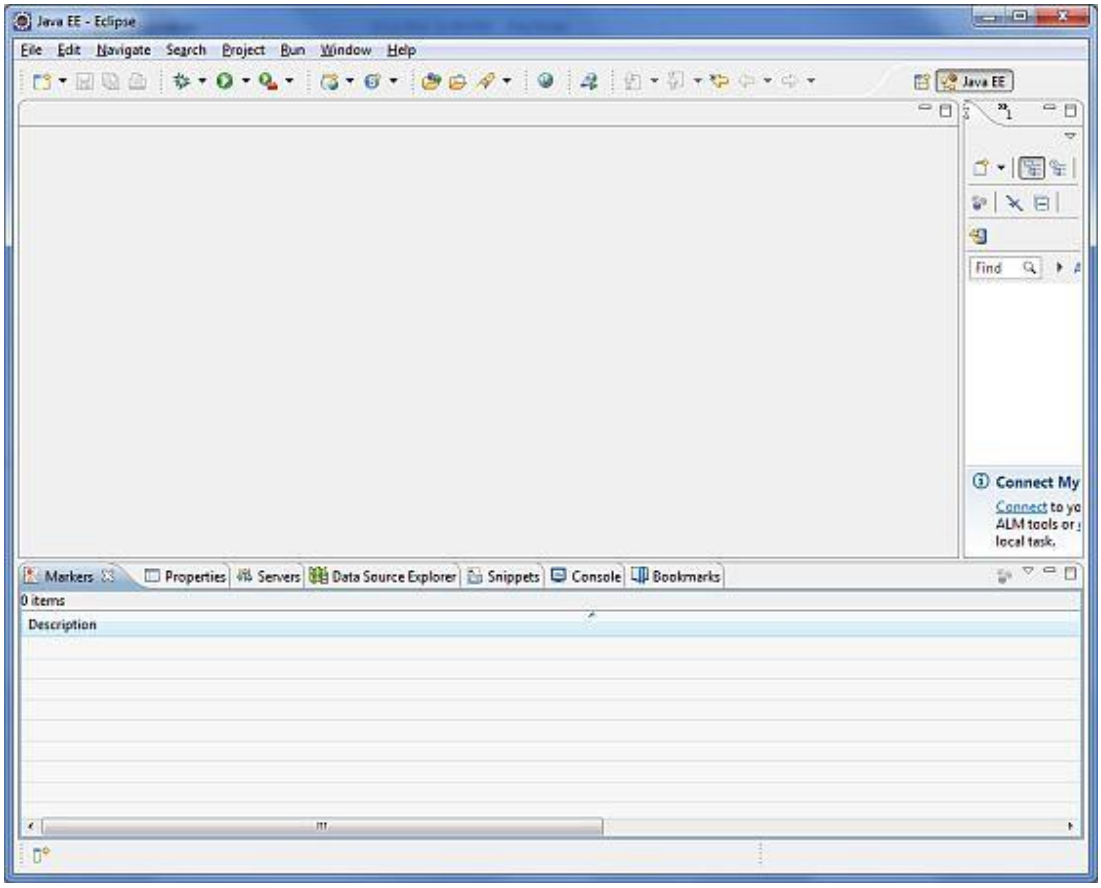


Η κύρια οθόνη του Διαχειριστή AVD δείχνει τις τρέχουσες εικονικές συσκευές.

2. Στην κύρια οθόνη του Διαχειριστή AVD κάνουμε κλικ στο **Create Virtual Device.**
3. Στο παράθυρο Select Hardware, επιλέγουμε κάποια συσκευή, όπως Nexus 6, και μετά πατάμε **Next.**
4. Επιλέγουμε την επιθυμητή έκδοση συστήματος για την AVD και πατάμε στο **Next**.
5. Κάνουμε έναν τελικό έλεγχο και πατάμε **Finish**.

Eclipse

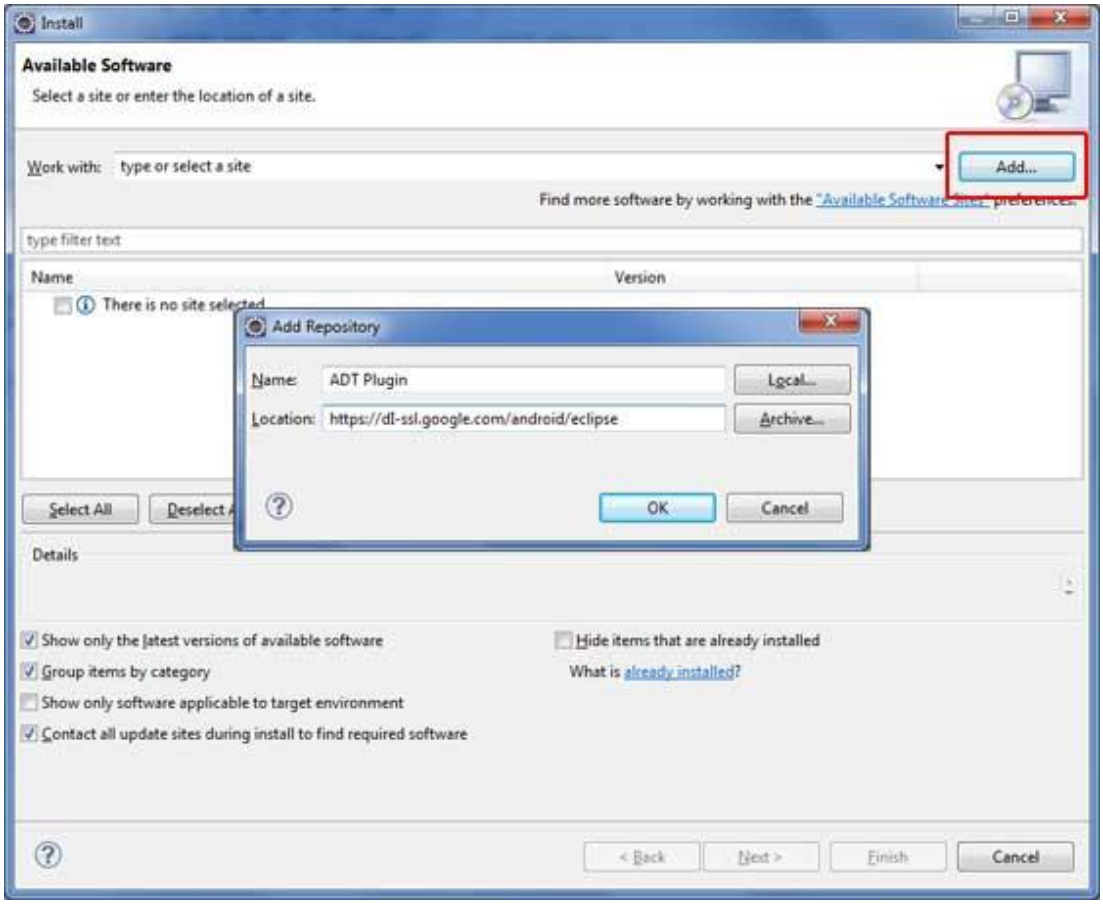
Το Android Studio διατίθεται και αυτό σε τρεις διαφορετικές εκδόσεις, για Windows, Linux, και Mac OS X.



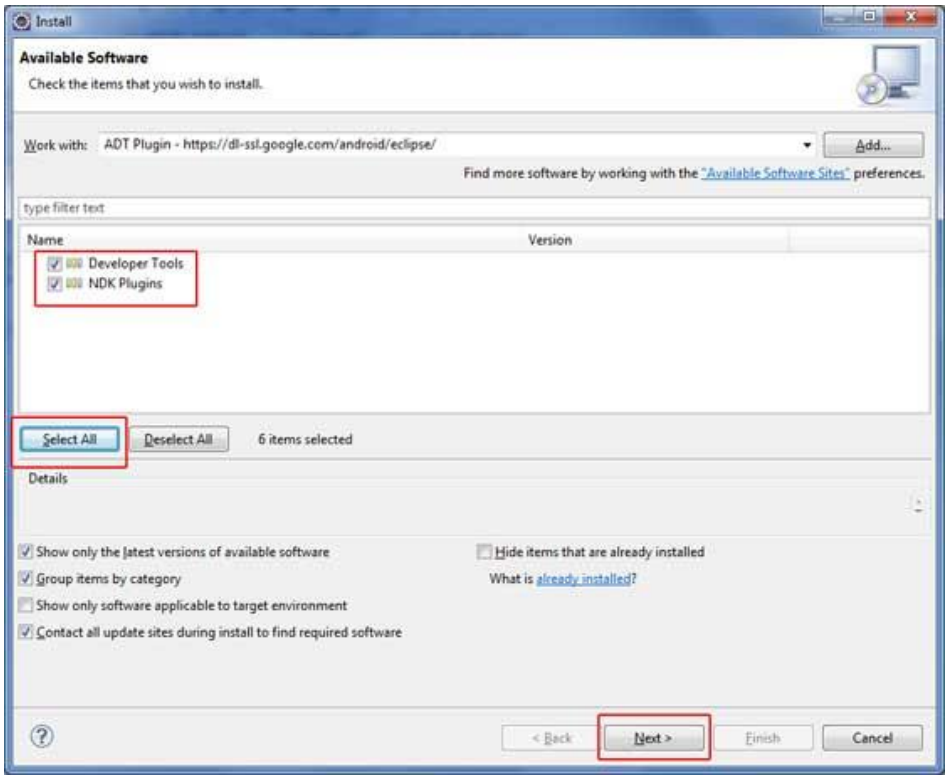
Android Development Tools (ADT) Plugin

Το ADT είναι ένα plugin για το Eclipse, το οποίο παρέχει πρόσβαση μέσω γραφικών διεπαφών χρήστη (GUI) σε διάφορα εργαλεία γραμμής εντολών του Android SDK, όπως επίσης και ένα εργαλείο για γρήγορη σχεδίαση, και κατασκευή των διεπαφών χρήστη μιας εφαρμογής.

Για να εγκαταστήσουμε το ADT θα πρέπει από το Eclipse να επιλέξουμε **Help > Install New Software**. Θα εμφανιστεί το ακόλουθο παράθυρο διαλόγου.



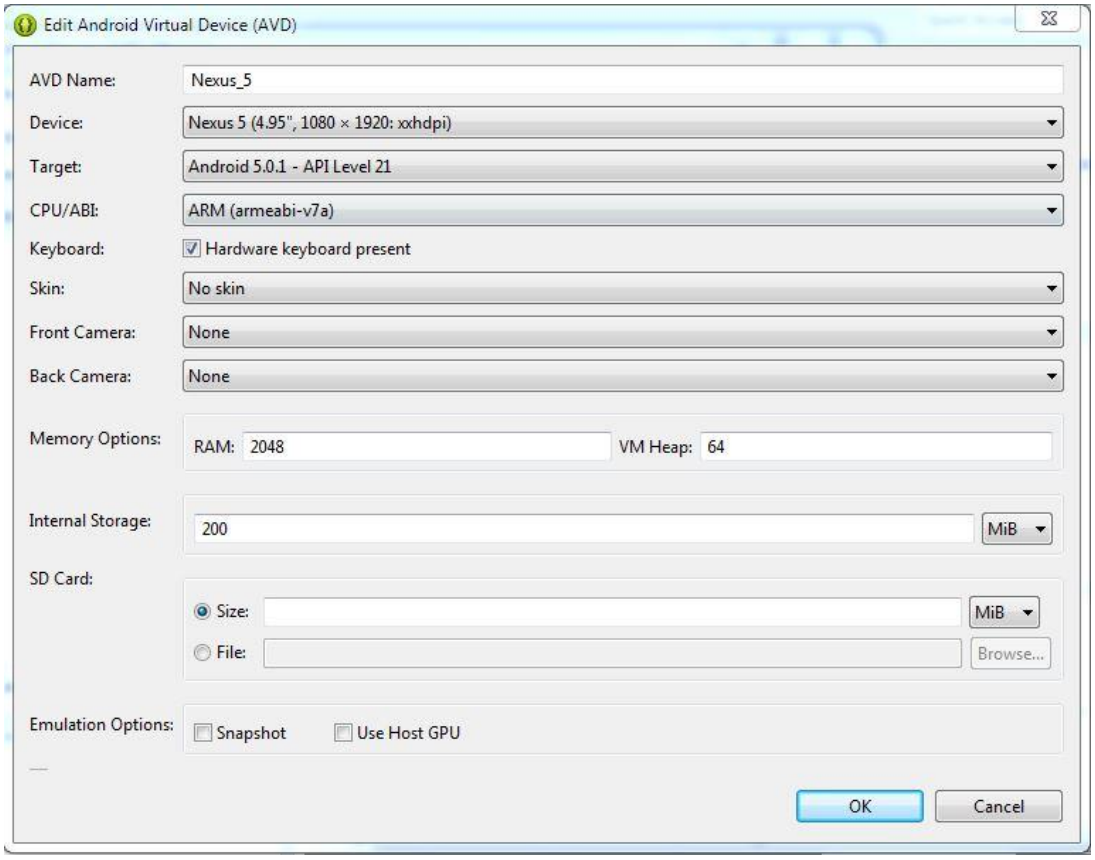
Κάνουμε κλικ στο κουμπί Add και βάζουμε ADT plugin σαν όνομα και *https://dl-ssl.google.com/android/eclipse/* σαν διεύθυνση. Το Eclipse θα εντοπίσει το plugin.



Επιλέγουμε **Select All** και **Next** για να συνεχίσουμε την εγκατάσταση.

Δημιουργία μιας εικονικής συσκευής Android (AVD)

Για την δημιουργία μιας AVD πρέπει να ανοίξουμε τον διαχειριστή AVD επιλέγοντας **Window > AVD Manager**. Επιλέγοντας **New** μπορούμε να φτιάξουμε μία καινούρια εικονική συσκευή, όπως φαίνεται στην ακόλουθη εικόνα.



Μετά την επιτυχή δημιουργία της AVD, το περιβάλλον προγραμματισμού είναι πλέον έτοιμο για την ανάπτυξη εφαρμογών για Android.

Διεπαφή Χρήστη (User Interface)

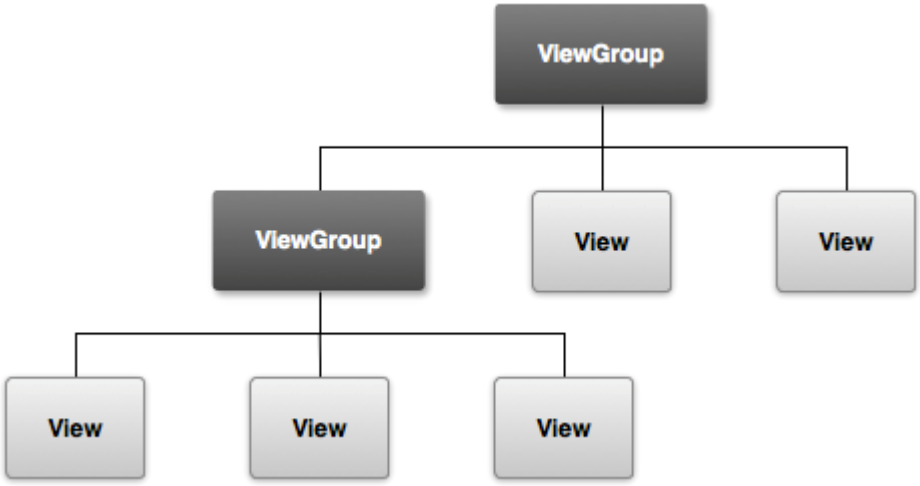
Επισκόπηση

Όλα τα στοιχεία μιας διεπαφής χρήστη σε μία εφαρμογή για Android φτιάχνονται χρησιμοποιώντας τα αντικείμενα View και ViewGroup. Ένα View είναι ένα αντικείμενο το οποίο σχεδιάζει κάτι στην οθόνη με το οποίο μπορεί να αλληλεπιδράσει ο χρήστης. Ένα ViewGroup είναι ένα αντικείμενο το οποίο κρατάει άλλα View (και ViewGroup) αντικείμενα προκειμένου να ορίσει την διάταξη μιας διεπαφής χρήστη.

Το Android παρέχει μία συλλογή από υποκλάσεις των κλάσεων View και ViewGroup οι οποίες προσφέρουν συνηθισμένα στοιχεία ελέγχου εισόδου (input controls), όπως κουμπιά και πεδία κειμένου, και διάφορους τύπους διατάξεων, όπως γραμμικές (linear) ή σχετικές διατάξεις (relative layouts).

Διάταξη Διεπαφής Χρήστη (User Interface Layout)

Η διεπαφή χρήστη ενός στοιχείου λογισμικού μιας εφαρμογής ορίζεται χρησιμοποιώντας μια ιεραρχία από αντικείμενα View και ViewGroup, όπως φαίνεται στην παρακάτω εικόνα. Κάθε ViewGroup είναι ένα αόρατο κοντέινερ το οποίο οργανώνει τα θυγατρικά Views, ενώ αυτά μπορεί να είναι στοιχεία ελέγχου εισόδου ή άλλα views τα οποία σχεδιάζουν κάποιο μέρος της διεπαφής χρήστη. Αυτό το ιεραρχικό δέντρο, μπορεί να είναι όσο απλό ή σύνθετο χρειάζεται (από πλευράς απόδοσης βέβαια μια απλή ιεραρχία είναι προτιμότερη).



Απεικόνιση μιας ιεραρχίας από views, η οποία ορίζει την διάταξη μιας διεπαφής χρήστη.

Για να ορίσουμε μία διάταξη, μπορούμε αν θέλουμε να δημιουργήσουμε αντικείμενα View προγραμματιστικά και να δημιουργήσουμε σταδιακά ένα δέντρο, αλλά ο ευκολότερος και πιο αποτελεσματικός τρόπος είναι με ένα αρχείο XML.

Το όνομα ενός στοιχείου XML για κάποιο View είναι αντίστοιχο της κλάσης που εκπροσωπεί. Έτσι, ένα στοιχείο <TextView> δημιουργεί ένα View τύπου TextView στη διεπαφή χρήστη, ενώ ένα στοιχείο <LinearLayout> δημιουργεί ένα ViewGroup τύπου LinearLayout.

Για παράδειγμα, μία απλή κάθετη διάταξη με ένα TextView και ένα Button είναι η ακόλουθη:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:orientation="vertical" >
```

```
    <TextView android:id="@+id/text"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="I am a TextView" />
    <Button android:id="@+id/button"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="I am a Button" />

</LinearLayout>
```

Όταν φορτώνεται μια διάταξη από ένα αρχείο XML της εφαρμογής μας, το Android δημιουργεί για κάθε κόμβο ένα αντικείμενο χρόνου εκτέλεσης (runtime object) το οποίο μπορούμε να χρησιμοποιήσουμε για να ορίσουμε πρόσθετες συμπεριφορές, να πάρουμε πληροφορίες για την τρέχουσα κατάστασή του, ή να τροποποιήσουμε την διάταξη.

Στοιχεία διεπαφής χρήστη (User Interface Components)

Δεν είναι απαραίτητο να φτιάξουμε ολόκληρη την διεπαφή χρήστη χρησιμοποιώντας αντικείμενα View και ViewGroup. Το Android παρέχει διάφορα στοιχεία λογισμικού τα οποία έχουν μία καθορισμένη διάταξη, για την οποία το μόνο που απαιτείται είναι να ορίσουμε το περιεχόμενο της. Το καθένα από αυτά τα στοιχεία διεπαφής χρήστη έχει το δικό του σετ από APIs τα οποία περιγράφονται στα αντίστοιχα έγγραφα. Παραδείγματα αποτελούν τα Action Bars, Dialogs και Status Notifications.

Διατάξεις (Layouts)

Οι διατάξεις ορίζουν την οπτική δομή μιας διεπαφής χρήστη, όπως είναι η διεπαφής χρήστη μιας Δραστηριότητας. Μπορούμε να δηλώσουμε μία διάταξη με δύο τρόπους:

- **Να δηλώσουμε τα στοιχεία μιας διεπαφής χρήστη σε XML:** Το Android παρέχει ένα λεξιλόγιο XML το οποίο αντιστοιχεί στις κλάσεις και υποκλάσεις της View.
- **Να δημιουργήσουμε τα στοιχεία της διάταξης κατά το χρόνο εκτέλεσης (runtime):** Μία εφαρμογή μπορεί να δημιουργήσει αντικείμενα View και ViewGroup (και να χειριστεί τις ιδιότητές τους) προγραμματιστικά.

Το framework του Android μας δίνει τη δυνατότητα να χρησιμοποιήσουμε κάποια από τις δύο ή και τις δύο από αυτές τις μεθόδους για να δηλώσουμε και να διαχειριστούμε τη διεπαφή χρήστη της εφαρμογής μας. Για παράδειγμα, θα μπορούσαμε να δηλώσουμε τις προεπιλεγμένες διατάξεις (default layouts) της εφαρμογής μας σε XML, συμπεριλαμβανομένων των στοιχείων που εμφανίζονται στην οθόνη και των ιδιοτήτων τους. Μετά μπορούμε να προσθέσουμε κώδικα στην εφαρμογή μας που να τροποποιεί την κατάσταση των αντικειμένων που εμφανίζονται στην οθόνη, συμπεριλαμβανομένων αυτών που έχουν δηλωθεί σε XML, κατά το χρόνο εκτέλεσης.

Το όφελος που έχουμε δηλώνοντας τη διεπαφή χρήστη σε XML είναι ότι διαχωρίζουμε καλύτερα με αυτό τον τρόπο την εμφάνιση της εφαρμογής από τον κώδικα που ελέγχει τη συμπεριφορά της. Οι περιγραφές σε XML της διεπαφής χρήστη δεν αποτελούν μέρος του κώδικα της εφαρμογής, το οποίο σημαίνει ότι μπορούμε να τις τροποποιήσουμε ή να τις προσαρμόσουμε χωρίς να χρειάζεται να τροποποιήσουμε τον πηγαίο κώδικα και να κάνουμε ξανά μεταγλώττιση. Για παράδειγμα, μπορούμε να φτιάξουμε διατάξεις σε XML για διαφορετικούς προσανατολισμούς (orientations) της οθόνης, για οθόνες διαφορετικών μεγεθών, και για διαφορετικές γλώσσες. Επιπλέον, δηλώνοντας τη διάταξη σε XML, απεικονίζεται πιο καθαρά η δομή της διεπαφής χρήστη, οπότε είναι πιο εύκολη η αποσφαλμάτωση.

Γενικά, το λεξιλόγιο της XML για τη δήλωση στοιχείων της διεπαφής χρήστη ακολουθεί τη δομή και την ονοματολογία των κλάσεων και των μεθόδων, όπου τα ονόματα των στοιχείων της XML αντιστοιχούν σε ονόματα κλάσεων, και τα ονόματα των ιδιοτήτων αντιστοιχούν σε μεθόδους.

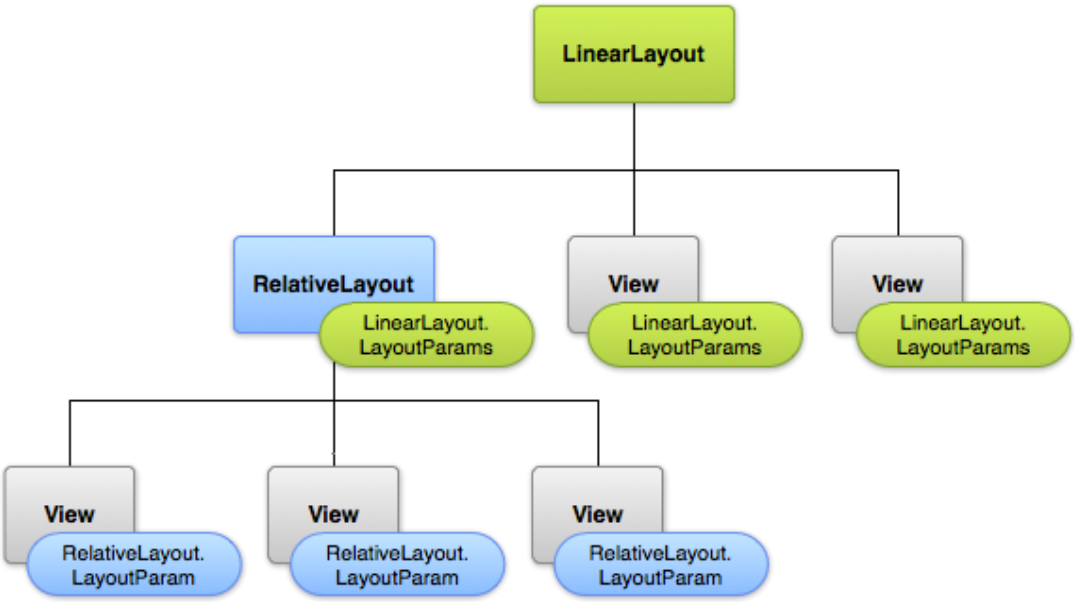
Χρησιμοποιώντας το λεξιλόγιο της XML για το Android, μπορούμε να σχεδιάσουμε γρήγορα διατάξεις για τις διεπαφές χρήστη, με τον ίδιο τρόπο που δημιουργούμε σελίδες σε HTML – με μια σειρά από ένθετα στοιχεία (nested elements).

Κάθε διάταξη πρέπει να περιλαμβάνει ακριβώς ένα ριζικό στοιχείο (root element), το οποίο πρέπει να είναι ένα αντικείμενο View ή ViewGroup. Αφού ορίσουμε το ριζικό στοιχείο, μπορούμε να προσθέσουμε επιπλέον αντικείμενα τύπου View ή ViewGroup σαν θυγατρικά στοιχεία προκειμένου να φτιάξουμε σταδιακά μια ιεραρχία αντικειμένων View η οποία ορίζει τη διάταξη.

Αφού ορίσουμε τη διάταξη σε XML, θα πρέπει να σώσουμε το xml αρχείο στο φάκελο res/layout/ έτσι ώστε να γίνει σωστά η μεταγλώττιση.

Ιδιότητες

Κάθε αντικείμενο View ή ViewGroup υποστηρίζει τις δικές του ιδιότητες XML. Ορισμένες ιδιότητες υποστηρίζονται από ένα τύπο αντικείμενου View (για παράδειγμα, το TextView υποστηρίζει την ιδιότητα textSize), αλλά αυτές οι ιδιότητες κληρονομούνται από κάθε αντικείμενο που επεκτείνει αυτή τη κλάση. Ορισμένες είναι κοινές σε όλα τα αντικείμενα View, γιατί κληρονομούνται από την ριζική κλάση View (όπως είναι η ιδιότητα id). Και, άλλες ιδιότητες θεωρούνται «παράμετροι διατάξεων», οι οποίες καθορίζονται από το μητρικό αντικείμενο ViewGroup.



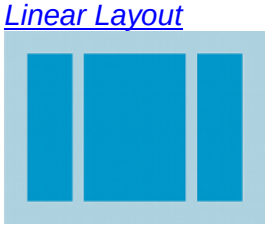
Κάθε θυγατρικό στοιχείο ορίζει κάποιες παραμέτρους διάταξης οι οποίες είναι κατάλληλες για το μητρικό στοιχείο, ενώ μπορεί επίσης να ορίζει διαφορετικές παραμέτρους διάταξης για τα δικά του θυγατρικά στοιχεία, όπως φαίνεται στην προηγούμενη εικόνα.

Συνηθισμένες Διατάξεις

Κάθε υποκλάση της κλάσης ViewGroup παρέχει ένα μοναδικό τρόπο για την απεικόνιση των αντικειμένων View που είναι ενσωματωμένα (nested) σε αυτήν. Παρακάτω αναφέρονται ορισμένοι από τους πιο συνηθισμένους τύπους διατάξεων οι οποίοι παρέχονται από την πλατφόρμα Android.

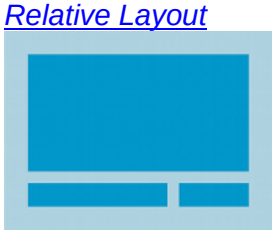
Σημείωση: Μολονότι μπορούμε να ενσωματώσουμε μία η περισσότερες διατάξεις εντός μιας άλλης διάταξης για να πετύχουμε τον επιθυμητό σχεδιασμό της διεπαφής χρήστη, θα πρέπει η ιεραρχία της διάταξης να είναι όσο το δυνατόν πιο «ρηχή». Η διάταξη θα σχεδιαστεί γρηγορότερα αν έχει λιγότερες ενσωματωμένες διατάξεις.

Linear Layout



Μία διάταξη η οποία οργανώνει τα στοιχεία σε μία οριζόντια ή κάθετη γραμμή. Δημιουργεί μία γραμμή κύλισης αν το μέγεθος του παραθύρου είναι μεγαλύτερο από το μέγεθος της οθόνης.

Relative Layout



Μας επιτρέπει να ορίσουμε την θέση των θυγατρικών στοιχείων σε σχέση με άλλα θυγατρικά στοιχεία ή σε σχέση με το μητρικό στοιχείο.

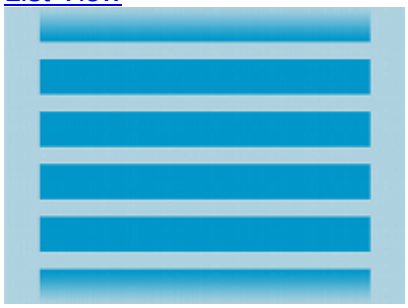
Web View

Χρησιμοποιείται για την εμφάνιση ιστοσελίδων.

Κατασκευή Διατάξεων με χρήση Αντάπτορα

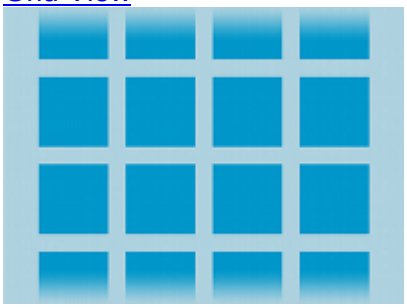
Όταν το περιεχόμενο της διάταξης είναι δυναμικό ή δεν είναι προκαθορισμένο, μπορούμε να χρησιμοποιήσουμε μία διάταξη η οποία είναι υποκλάση της κλάσης AdapterView προκειμένου να γεμίσουμε την διάταξη με Views κατά το χρόνο εκτέλεσης (runtime). Μία υποκλάση της κλάσης AdapterView χρησιμοποιεί έναν Αντάπτορα (Adapter) για να συνδέσει τα δεδομένα με τη διάταξή της. Ο αντάπτορας συμπεριφέρεται σαν μεσάζων ανάμεσα στα δεδομένα και στη διάταξη τύπου AdapterView – ο Αντάπτορας παίρνει τα δεδομένα (από μία πηγή όπως είναι ένας πίνακας ή από μία βάση δεδομένων) και μετατρέπει την κάθε καταχώριση σε ένα View το οποίο μπορεί να εισαχθεί στη διάταξη. Στις συνηθισμένες διατάξεις που υποστηρίζονται από κάποιον αντάπτορα περιλαμβάνονται:

List View



Εμφανίζει μία κυλιόμενη λίστα η οποία περιλαμβάνει μία κα μοναδική στήλη.

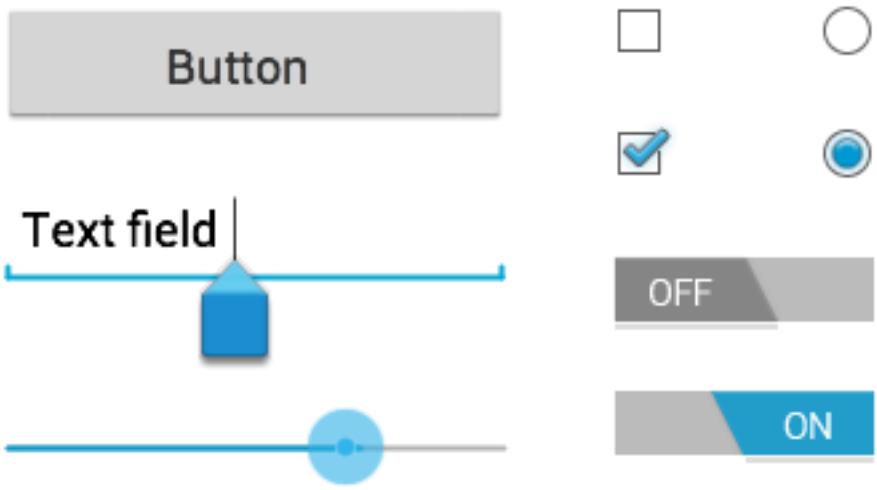
Grid View



Εμφανίζει ένα κυλιόμενο πλέγμα γραμμών και στηλών.

Στοιχεία ελέγχου εισόδου (Input Controls)

Τα στοιχεία ελέγχου εισόδου είναι τα διαδραστικά στοιχεία της διεπαφής χρήστη μιας εφαρμογής. Το Android παρέχει μία μεγάλη ποικιλία στοιχείων τα οποία μπορούμε να χρησιμοποιήσουμε στη διεπαφή χρήστη, όπως κουμπιά, πεδία κειμένου, γραμμές αναζήτησης (seek bars), πλαίσια ελέγχου (checkboxes), κουμπιά για ζουμ, κουμπιά εναλλαγής (toggle buttons), και πολλά άλλα. Κάθε στοιχείο ελέγχου εισόδου υποστηρίζει μία σειρά από γεγονότα εισόδου (input events) έτσι ώστε να μπορούμε να χειριστούμε γεγονότα όπως όταν ο χρήστης εισάγει κάποιο κείμενο ή ακουμπάει κάποιο κουμπί.



Συνηθισμένα στοιχεία ελέγχου εισόδου

Ακολουθεί μία λίστα με ορισμένα συνηθισμένα στοιχεία τα οποία μπορούν να χρησιμοποιηθούν σε μία εφαρμογή.

Σημείωση: Το Android παρέχει αρκετά περισσότερα στοιχεία από αυτά που αναφέρονται εδώ. Μπορούμε επίσης να φτιάξουμε προσαρμοσμένα στοιχεία (custom components), αν τα υπάρχοντα δεν μας καλύπτουν για την εφαρμογή που αναπτύσσουμε.

Τύπος στοιχείου	**Περιγραφή**
Κουμπί	Ένα κουμπί το οποίο ο χρήστης μπορεί να πατήσει, ή να κάνει κλικ, για να εκτελέσει κάποια ενέργεια.
Πεδίο κειμένου	Χρησιμοποιείται για εισαγωγή κειμένου.
Πλαίσιο ελέγχου	Ένας on/off διακόπτης ο οποίος μπορεί να αλλάξει από τον χρήστη. Τα πλαίσια ελέγχου θα πρέπει να χρησιμοποιούνται όταν παρουσιάζουμε στον χρήστη επιλογές οι οποίες δεν είναι αλληλοαποκλειόμενες.
Κουμπί επιλογής (radio button)	Παρόμοιο με τα πλαίσια ελέγχου, με τη διαφορά ότι οι επιλογές είναι αμοιβαία αποκλειόμενες.
Κουμπί εναλλαγής	Ένα κουμπί on/off με ένδειξη φανού (light indicator).
Spinner	Μία αναπτυσσόμενη λίστα η οποία επιτρέπει στον χρήστη να επιλέξει μία τιμή από κάποιο σύνολο.
Επιλογέας (Picker)	Ένα παράθυρο διαλόγου το οποίο επιτρέπει στον χρήστη να επιλέξει κάποια ημερομηνία ή ώρα.

Γεγονότα Εισόδου (Input Events)

Στο Android, υπάρχουν παραπάνω του ενός τρόποι για να συλλάβουμε τα γεγονότα από την αλληλεπίδραση ενός χρήστη με την εφαρμογή. Όταν έχουμε να κάνουμε με γεγονότα εντός της διεπαφής χρήστη, η προσέγγιση είναι να συλλάβουμε (capture) τα γεγονότα από το αντικείμενο View με το οποίο αλληλεπιδρά ο χρήστης. Η κλάση View παρέχει τα μέσα για να το κάνουμε αυτό.

Εντός των διαφόρων κλάσεων View τις οποίες χρησιμοποιούμε για να συνθέσουμε μια διάταξη, μπορεί να παρατηρήσουμε αρκετές δημόσιες μεθόδους callback οι οποίες φαίνονται χρήσιμες για σύλληψη γεγονότων εισόδου. Αυτές οι μέθοδοι καλούνται από το framework του Android όταν η αντίστοιχη ενέργεια συμβεί σε αυτό το αντικείμενο. Για παράδειγμα, όταν αγγίξουμε ένα View (όπως ένα κουμπί), η μέθοδος onTouchEvent() καλείται σε αυτό το αντικείμενο. Ωστόσο, προκειμένου να το συλλάβουμε αυτό, θα πρέπει να επεκτείνουμε την κλάση και να υπερβούμε (override) την μέθοδο. Ωστόσο, το να επεκτείνουμε κάθε αντικείμενο View για να χειριστούμε ένα τέτοιο γεγονός δεν θα ήταν πρακτικό. Αυτός είναι ο λόγος για τον οποίο η κλάση View περιέχει επίσης μία συλλογή από ενσωματωμένες διασυνδέσεις (nested Interfaces) με μεθόδους callback τις οποίες μπορούμε πολύ πιο εύκολα να ορίσουμε. Αυτές οι διασυνδέσεις, οι οποίες ονομάζονται ανιχνευτές γεγονότων (event listeners), είναι το εισιτήριο μας για να συλλάβουμε την αλληλεπίδραση του χρήστη με την διεπαφή χρήστη.

Μολονότι χρησιμοποιούνται πιο συχνά οι ανιχνευτές γεγονότων για τη σύλληψη γεγονότων, υπάρχουν περιπτώσεις όπου θέλουμε να επεκτείνουμε μία κλάση View, προκειμένου να φτιάξουμε ένα προσαρμοσμένο στοιχείο (custom component). Σε αυτήν την περίπτωση, μπορούμε να ορίσουμε την προεπιλεγμένη συμπεριφορά των στοιχείων για τα γεγονότα εισόδου χρησιμοποιώντας τους χειριστές γεγονότων (event handlers) (π.χ. onTouchEvent()).

Ανιχνευτές Γεγονότων

Ο ανιχνευτής γεγονότων είναι μία διασύνδεση στην κλάση View που περιέχει μία και μοναδική μέθοδο callback. Αυτές οι μέθοδοι θα κληθούν από το framework του Android όταν ο χρήστης αλληλεπιδράσει με το View με το οποίο έχει συνδεθεί ο ανιχνευτής γεγονότων.

Στις διασυνδέσεις των ανιχνευτών γεγονότων περιλαμβάνονται οι ακόλουθες μέθοδοι callback.

onClick()

> Από το View.OnClickListener. Καλείται είτε όταν ο χρήστης ακουμπήσει το αντικείμενο, είτε όταν εστιάσει (focus) στο αντικείμενο με τα κουμπιά πλοήγησης ή το trackball και πατήσει το κατάλληλο κουμπί “enter” ή πατήσει το trackball.

onLongClick()

Από το View.OnLongClickListener. Καλείται είτε όταν ο χρήστης ακουμπήσει το αντικείμενο, είτε όταν εστιάσει (focus) στο αντικείμενο με τα κουμπιά πλοήγησης ή το trackball και πατήσει το κατάλληλο κουμπί “enter” ή πατήσει το trackball για ένα δευτερόλεπτο.

onFocusChange()

Από το View.OnFocusChangeListener. Καλείται όταν ο χρήστης χρησιμοποιήσει τα πλήκτρα πλοήγησης ή το trackball για να εστιάσει σε κάποιο αντικείμενο ή όταν αυτό χάσει την εστίαση.

onKey()

Από το View.OnKeyListener. Καλείται όταν ο χρήστης πατήσει ή αφήσει κάποιο κουμπί (hardware key) ενώ έχει εστιάσει σε ένα αντικείμενο.

onTouch()

Από το View.OnTouchListener. Καλείται όταν προκληθεί από τον χρήστη κάποιο συμβάν αφής (touch event). Συμβάν αφής έχουμε όταν ο χρήστης πατήσει, αφήσει ή κάνει κάποια άλλη κίνηση πάνω στην οθόνη (στα όρια του αντικειμένου).

onCreateContextMenu()

Από το View.OnCreateContextMenuListener. Καλείται όταν εμφανιστεί κάποιο Context Menu. Context Menu μπορεί να εμφανιστεί μετά από ένα παρατεταμένο κλικ (long click).

Αυτές είναι οι μοναδικές μέθοδοι που περιλαμβάνουν οι αντίστοιχες διασυνδέσεις. Προκειμένου να ορίσουμε κάποια από αυτές τις μεθόδους και να χειριστούμε αυτά τα γεγονότα, θα πρέπει να υλοποιήσουμε την διασυνδέση μέσα στη Δραστηριότητα ή να την ορίσουμε σαν ανώνυμη κλάση. Μετά, περνάμε ένα στιγμιότυπο της υλοποίησής μας στην αντίστοιχη View.set...Listener() μέθοδο. (Π.χ. καλούμε την setOnClickListener() και της περνάμε την υλοποίηση του OnClickListener).

Μενού

Τα μενού είναι ένα συνηθισμένο στοιχείο της διεπαφής χρήστη σε πολλούς τύπους εφαρμογών.

Από το Android 3.0 (API level 11) και έπειτα, μία συσκευή Android δεν είναι απαραίτητο να διαθέτει κουμπί Menu. Με αυτήν την αλλαγή, οι εφαρμογές δεν θα πρέπει να βασίζονται στο παραδοσιακό εξαθέσιο μενού, και αντ’ αυτού πρέπει να παρέχουν μία μπάρα ενεργειών (action bar) στην οποία θα υπάρχουν επιλογές για συνηθισμένες ενέργειες που κάνουν οι χρήστες.

Υπάρχουν τρεις βασικοί τύποι μενού στο Android:

Μενού επιλογών και μπάρα ενεργειών

Το μενού επιλογών αποτελεί την βασική συλλογή στοιχείων μενού μιας Δραστηριότητας. Εδώ μπορούν να τοποθετηθούν ενέργειες όπως «Αναζήτηση», «Δημιουργία email», και «Ρυθμίσεις».

Αν αναπτύσσουμε μία εφαρμογή για την 2.3 έκδοση του Android ή κάποια παλιότερη, οι χρήστες μπορούν να δουν αυτές τις επιλογές πατώντας το κουμπί Menu.

Από την έκδοση 3.0 και έπειτα, τα στοιχεία του μενού επιλογών περιέχονται στην μπάρα ενεργειών. Ορισμένα από αυτά μπορεί να είναι τοποθετημένα στην περιοχή υπερχείλισης (overflow).

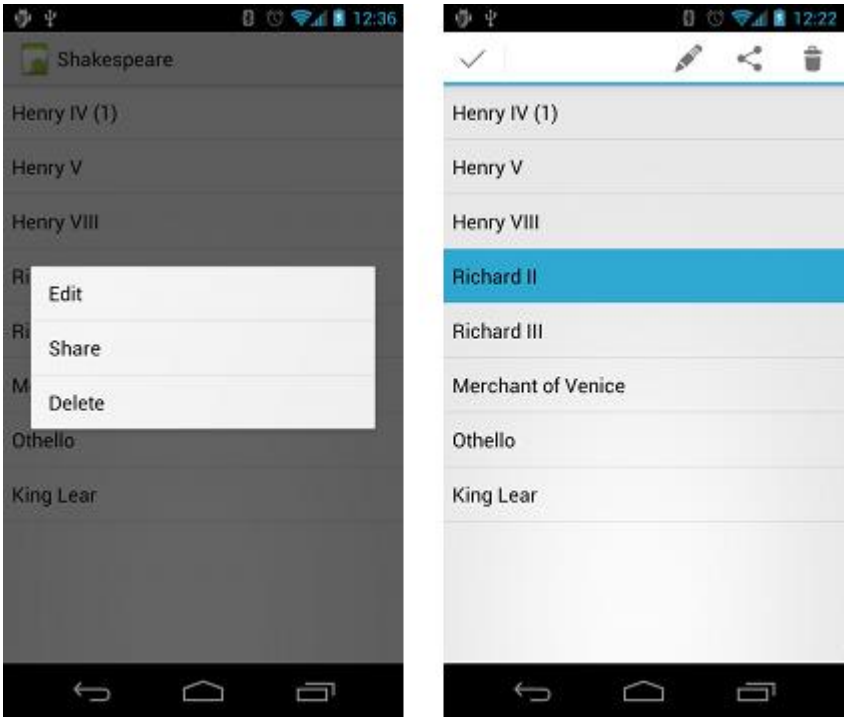
Μία μπάρα ενεργειών η οποία περιλαμβάνει [1] το εικονίδιο της εφαρμογής, [2] δύο στοιχεία ενεργειών (action items), και [3] την υπερχείλιση ενεργειών (action overflow).

Μενού επιλογών στον Browser, στην έκδοση 2.3 του Android.

Μενού περιβάλλοντος (context menu) και λειτουργία ενεργειών περιβάλλοντος (contextual action mode)

Το μενού περιβάλλοντος είναι ένα αιωρούμενο μενού το οποίο εμφανίζεται όταν ο χρήστης πραγματοποιήσει παρατεταμένο κλικ πάνω σε κάποιο στοιχείο. Παρέχει ενέργειες οι οποίες αφορούν το περιεχόμενο του στοιχείου.

Όταν αναπτύσσουμε για την έκδοση 3.0 ή κάποια νεότερη, είναι προτιμότερο να χρησιμοποιούμε την λειτουργία ενεργειών περιβάλλοντος. Αυτή η λειτουργία εμφανίζει στοιχεία ενεργειών (action items) σε μία μπάρα στην κορυφή της οθόνης και επιτρέπει στον χρήστη να επιλέξει ταυτόχρονα πολλά από τα στοιχεία που εμφανίζονται στην οθόνη.



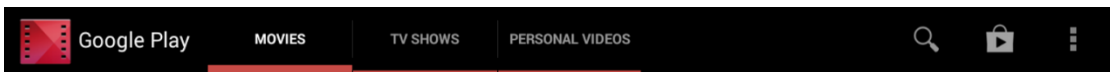
Εικόνες ενός αιωρούμενου μενού περιβάλλοντος (αριστερά) και της μπάρας ενεργειών περιβάλλοντος (δεξιά).

Αναδυόμενο μενού (Popup menu)

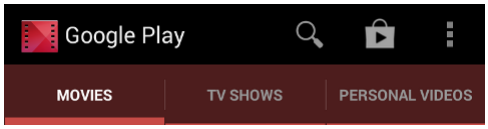
Ένα αναδυόμενο μενού εμφανίζει μία λίστα στοιχείων σε μία κάθετη λίστα η οποία είναι αγκυρωμένη στο view από το οποίο εμφανίστηκε το μενού. Είναι κατάλληλο όταν θέλουμε να εμφανίσουμε μια σειρά από ενέργειες οι οποίες σχετίζονται με συγκεκριμένο περιεχόμενο (π.χ. επιλογές «προώθησης» ή «απάντησης» για κάποιο email) ή για την εμφάνιση επιλογών για το δεύτερο μέρος μιας εντολής. Οι ενέργειες ενός αναδυόμενου μενού δεν πρέπει να επηρεάζουν άμεσα το περιεχόμενο του view – για αυτόν τον σκοπό υπάρχει το μενού ενεργειών περιβάλλοντος που αναφέρθηκε προηγουμένως.

Μπάρα Ενεργειών (Action Bar)

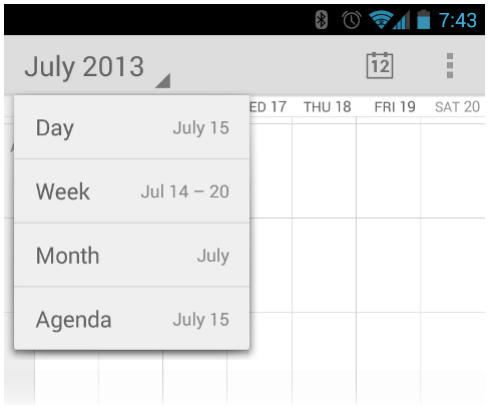
Η μπάρα ενεργειών αποτελεί χαρακτηριστικό ενός παραθύρου. Πληροφορεί το χρήστη για την περιοχή της εφαρμογής στην οποία βρίσκεται, εμφανίζει μία λίστα σημαντικών ενεργειών (π.χ. «Αναζήτηση»), και επιπλέον παρέχει καρτέλες και αναδυόμενες λίστες για την πλοήγηση του χρήστη στην εφαρμογή.



Μπάρα ενεργειών με καρτέλες σε ευρεία οθόνη.



Καρτέλες, τοποθετημένες χαμηλότερα, σε οθόνη μικρότερου εύρους.

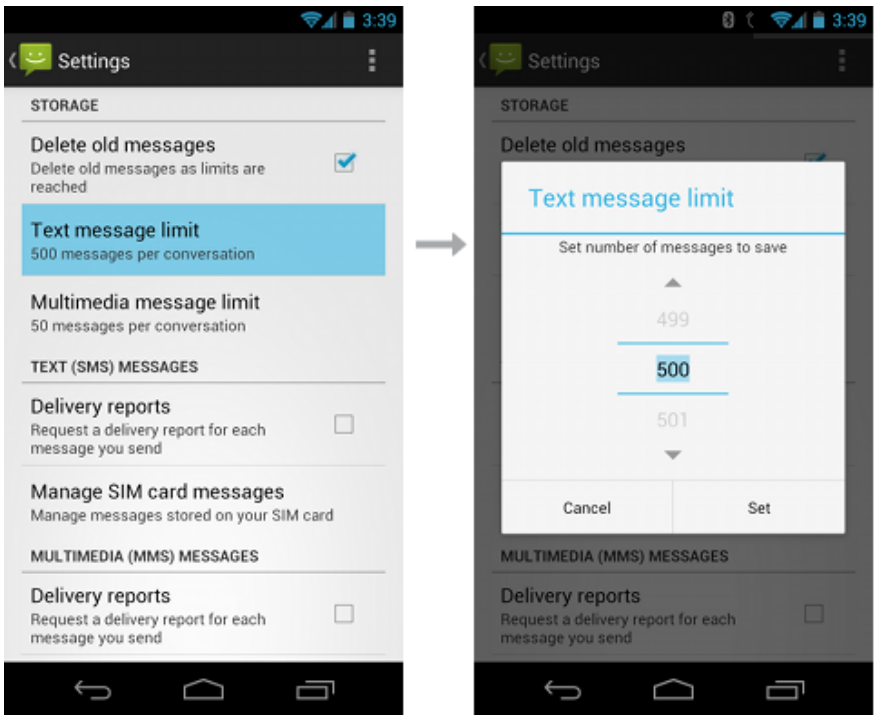


Μπάρα ενεργειών με αναδυόμενη λίστα πλοήγησης.

Ρυθμίσεις

Οι εφαρμογές συχνά περιλαμβάνουν ρυθμίσεις οι οποίες επιτρέπουν στους χρήστες να τροποποιούν τα χαρακτηριστικά και τη συμπεριφορά μιας εφαρμογής. Για παράδειγμα, ορισμένες εφαρμογές επιτρέπουν στους χρήστες να καθορίσουν το αν οι ειδοποιήσεις είναι ενεργοποιημένες ή το πόσο συχνά συγχρονίζει η εφαρμογή τα δεδομένα με το cloud.

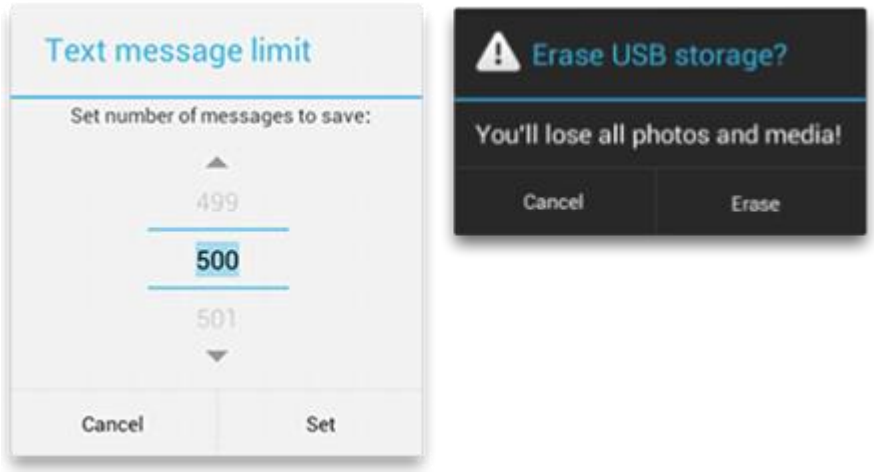
Αν θέλουμε να παρέχει κάποιες ρυθμίσεις η εφαρμογή μας, προτείνεται να γίνει χρήση των Preference APIs του Android, έτσι ώστε να φτιάξουμε μια διεπαφή χρήστη η οποία θα είναι παρόμοια με αυτές που χρησιμοποιούν άλλες εφαρμογές (συμπεριλαμβανομένων των ρυθμίσεων του συστήματος).



Εικόνες από τις ρυθμίσεις του Android Messaging. Επιλέγοντας ένα στοιχείο το οποίο ορίζεται από ένα Preference, ανοίγει μία διεπαφή χρήστη προκειμένου να αλλάξει η ρύθμιση.

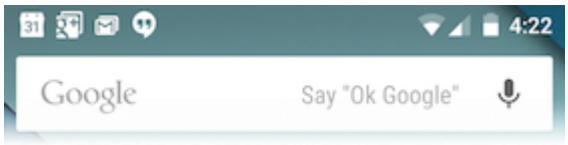
Παράθυρο Διαλόγου (Dialog)

Ένα παράθυρο διαλόγου είναι ένα μικρό παράθυρο το οποίο προτρέπει τον χρήστη να κάνει μία επιλογή ή να εισαγάγει επιπλέον πληροφορίες. Δεν καταλαμβάνει ολόκληρη την οθόνη, και επιπλέον απαιτείται από τους χρήστες να κάνουν μία ενέργεια πριν μπορέσουν να συνεχίσουν.

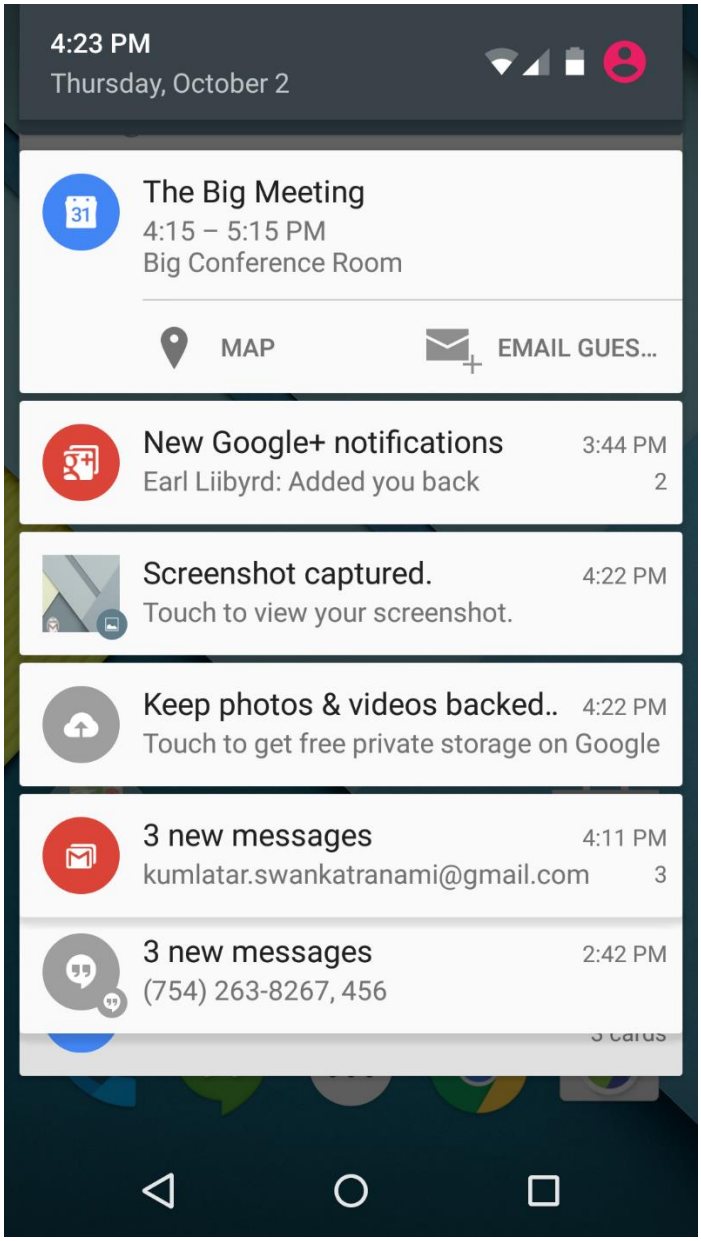


Ειδοποιήσεις

Η ειδοποίηση είναι ένα μήνυμα το οποίο μπορούμε να εμφανίσουμε στον χρήστη έξω από το περιβάλλον εργασίας χρήστη (user interface) της εφαρμογής. Όταν λέμε στο σύστημα να δημιουργήσει μία ειδοποίηση, αυτή εμφανίζεται πρώτα σαν εικονίδιο στην περιοχή ειδοποιήσεων (notification area) στο πάνω μέρος της οθόνης. Για να δει τις λεπτομέρειες της ειδοποίησης, ο χρήστης ανοίγει τον σχεδιαστή ειδοποιήσεων (notification drawer). Η περιοχή ειδοποιήσεων και ο σχεδιαστής ειδοποιήσεων είναι περιοχές που ελέγχονται από το σύστημα στις οποίες ο χρήστης έχει πρόσβαση οποτεδήποτε.



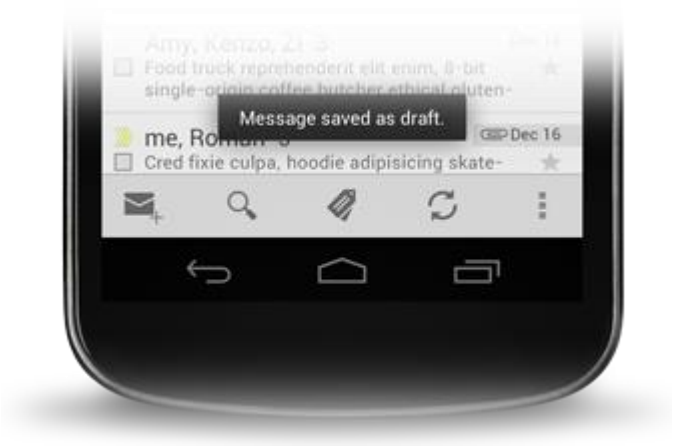
Ειδοποιήσεις στην περιοχή ειδοποιήσεων.



Οι ειδοποιήσεις όπως εμφανίζονται στον σχεδιαστή ειδοποιήσεων.

Αναδυόμενες Ειδοποιήσεις (Toasts)

Οι αναδυόμενες ειδοποιήσεις δίνουν πληροφορίες για κάποια λειτουργία σε ένα μικρό αναδυόμενο παράθυρο. Καταλαμβάνουν μόνο τον χώρο που απαιτείται για το μήνυμα. Η τρέχουσα δραστηριότητα παραμένει ενεργή και ο χρήστης μπορεί να αλληλεπιδράσει με αυτήν. Για παράδειγμα, αν φύγουμε από κάποιο email πριν το στείλουμε, θα εμφανιστεί μία αναδυόμενη ειδοποίηση με το μήνυμα “Draft saved” η οποία μας ειδοποιεί ότι μπορούμε να συνεχίσουμε την επεξεργασία αργότερα. Οι αναδυόμενες ειδοποιήσεις εξαφανίζονται αυτόματα μετά από ένα μικρό χρονικό διάστημα.



2D και 3D Γραφικά

Όταν γράφουμε μία εφαρμογή, είναι σημαντικό να προσδιορίσουμε σωστά τις γραφικές απαιτήσεις της. Διαφορετικές εργασίες γραφικών απαιτούν τη χρήση μιας ποικιλίας τεχνικών για να ολοκληρωθούν κατά τον βέλτιστο τρόπο. Για παράδειγμα, τα γραφικά και τα animations για μία αρκετά στατική εφαρμογή θα πρέπει να υλοποιηθούν πολύ διαφορετικά από τα γραφικά και τα animations ενός διαδραστικού παιχνιδιού. Εδώ, θα αναφερθούν ορισμένες από τις επιλογές που υπάρχουν για την σχεδίαση γραφικών στο Android, όπως επίσης και οι εργασίες για τις οποίες είναι καταλληλότερες.

Canvas και Drawables

Τα framework APIs του Android παρέχουν ένα σύνολο από APIs για 2D σχεδίαση τα οποία δίνουν τη δυνατότητα να αποδώσουμε (render) τα δικά μας προσαρμοσμένα γραφικά (custom graphics) σε ένα Canvas ή να τροποποιήσουμε υπάρχοντα Views για να προσαρμόσουμε την εμφάνισή τους και τη συμπεριφορά τους. Όταν σχεδιάζουμε 2D γραφικά, συνήθως το κάνουμε με έναν από τους ακόλουθους δύο τρόπους:

α. Σχεδιάζουμε τα γραφικά ή τα animations σε ένα αντικείμενο View από τη διάταξη. Κατ' αυτόν τον τρόπο, η σχεδίαση των γραφικών γίνεται με βάση την διαδικασία σχεδίασης του συστήματος για μία ιεραρχία από Views – απλώς ορίζουμε τα γραφικά που θα τοποθετηθούν μέσα στο View.

β. Σχεδιάζουμε τα γραφικά σε ένα Canvas. Κατ' αυτόν τον τρόπο, καλούμε την μέθοδο onDraw της κατάλληλης κλάσης (στην οποία περνιέται το Canvas), ή μία από τις μεθόδους draw...() του Canvas (όπως drawPicture()). Με αυτόν τον τρόπο, έχουμε τον έλεγχο του όποιου animation.

Η επιλογή «α», η σχεδίαση σε κάποιο View, αποτελεί την καλύτερη επιλογή όταν θέλουμε να σχεδιάσουμε απλά γραφικά τα οποία δεν χρειάζεται να αλλάζουν δυναμικά και δεν αποτελούν μέρος ενός απαιτητικού, από πλευράς απόδοσης, παιχνιδιού. Για παράδειγμα, θα πρέπει να σχεδιάζουμε τα γραφικά

σε ένα View όταν θέλουμε να εμφανίσουμε στατικά γραφικά ή προκαθορισμένα animation, σε μία κατά τα άλλα στατική εφαρμογή.

Η επιλογή «β», η σχεδίαση σε Canvas, είναι προτιμότερη όταν η εφαρμογή πρέπει να επανασχεδιάζεται συχνά. Εφαρμογές όπως βιντεοπαιχνίδια θα πρέπει να σχεδιάζουν με χρήση του Canvas. Ωστόσο, υπάρχουν παραπάνω του ενός τρόποι για να γίνει αυτό:

- Στο νήμα που μπορεί να τροποποιήσει την διεπαφή χρήστη (main thread ή UI thread), όπου δημιουργούμε ένα προσαρμοσμένο στοιχείο View στη διάταξή μας, καλούμε την invalidate(), και χειριζόμαστε την callback μέθοδο onDraw().
- Ή, σε ένα ξεχωριστό νήμα, όπου διαχειριζόμαστε ένα SurfaceView και σχεδιάζουμε στο Canvas όσο γρήγορα επιτρέπει το νήμα (δεν απαιτείται σε αυτήν την περίπτωση να καλέσουμε την invalidate()).

Σχεδίαση με χρήση Canvas

Όταν γράφουμε μία εφαρμογή στην οποία θέλουμε να κάνουμε εξειδικευμένη σχεδίαση ή/και να ελέγξουμε το animation των γραφικών, θα πρέπει να σχεδιάσουμε μέσω ενός Canvas. Το Canvas είναι για μας ένα είδος διασύνδεσης για την επιφάνεια στην οποία θα σχεδιαστούν τελικά τα γραφικά – κρατάει όλες τις εντολές σχεδίασης. Μέσω του Canvas, η σχεδίαση γίνεται τελικά σε ένα Bitmap, το οποίο τοποθετείται στο παράθυρο.

Σε περίπτωση που σχεδιάζουμε εντός της onDraw() callback, το Canvas μας παρέχεται και χρειάζεται απλώς να «τοποθετηθούν» οι εντολές σχεδίασης πάνω του. Μπορούμε επίσης να πάρουμε ένα Canvas από την SurfaceHolder.lockCanvas(), όταν εργαζόμαστε με ένα αντικείμενο SurfaceView. Ωστόσο, αν πρέπει να δημιουργήσουμε ένα νέο Canvas, τότε πρέπει να ορίσουμε το Bitmap πάνω στο οποίο θα γίνει τελικά η σχεδίαση. Το Bitmap είναι απαραίτητο για το Canvas. Μπορούμε να δημιουργήσουμε ένα νέο Canvas ως εξής:

```
Bitmap b = Bitmap.createBitmap(100, 100, Bitmap.Config.ARGB_8888);
Canvas c = new Canvas(b);
```

Έτσι το Canvas θα σχεδιάσει στο Bitmap που έχουμε ορίσει. Αφού σχεδιάσουμε πάνω του με το Canvas, μπορούμε να μεταφέρουμε το Bitmap σε άλλο Canvas με μία από τις μεθόδους Canvas.drawBitmap(Bitmap,...). Προτείνεται να σχεδιάσουμε τα τελικά γραφικά μέσω ενός Canvas το οποίο παρέχεται από την View.onDraw() or SurfaceHolder.lockCanvas().

Η κλάση Canvas έχει μια σειρά από μεθόδους σχεδίασης που μπορούμε να χρησιμοποιήσουμε, όπως drawBitmap(...), drawRect(...), drawText(...) και πολλές άλλες. Άλλες κλάσεις που ίσως χρησιμοποιήσουμε έχουν και αυτές μεθόδους draw(). Για παράδειγμα, μπορεί να έχουμε κάποια αντικείμενα Drawable τα οποία θέλουμε να τα βάλουμε στο Canvas. Η κλάση Drawable έχει τη δική της μέθοδο draw() που παίρνει το Canvas σαν παράμετρο.

Σχεδίαση σε View

Αν για την εφαρμογή μας δεν απαιτείται μεγάλη ποσότητα επεξεργασίας ή υψηλός ρυθμός καρέ (frame-rate) (ίσως για ένα παιχνίδι σκακιού, φιδάκι, ή κάποια άλλη εφαρμογή με αργό animation), τότε θα μπορούσαμε να δημιουργήσουμε ένα προσαρμοσμένο στοιχείο View (custom View) και να σχεδιάσουμε με ένα Canvas στην View.onDraw(Canvas canvas). Το πιο βολικό σε αυτήν την περίπτωση είναι ότι το framework του Android μας παρέχει ένα Canvas μέσω του οποίου μπορούμε να σχεδιάσουμε.

Σχεδίαση σε SurfaceView

Η SurfaceView είναι μια ειδική υποκλάση της View η οποία προσφέρει μια αποκλειστική επιφάνεια σχεδίασης. Ο στόχος είναι να προσφέρει αυτήν την αποκλειστική επιφάνεια σε ένα δευτερεύων νήμα της εφαρμογής, έτσι ώστε να μην χρειάζεται η εφαρμογή να περιμένει μέχρι η ιεραρχία των Views να είναι έτοιμη να σχεδιαστεί. Αντιθέτως, ένα δευτερεύων νήμα το οποίο έχει μία αναφορά σε ένα SurfaceView μπορεί να σχεδιάσει στο δικό του Canvas με το δικό του ρυθμό.

Drawables

Το Android προσφέρει μία προσαρμοσμένη βιβλιοθήκη 2D γραφικών για τον σχεδιασμό σχημάτων και εικόνων. Στο πακέτο android.graphics.drawable μπορούμε να βρούμε τις κλάσεις που χρησιμοποιούνται συχνά για την σχεδίαση 2D γραφικών.

Το Drawable είναι μία γενική αφαίρεση για «κάτι το οποίο μπορεί να σχεδιαστεί». Η κλάση Drawable επεκτείνεται έτσι ώστε να ορίσει διαφορετικά είδη drawable γραφικών, συμπεριλαμβανομένων των BitmapDrawable, ShapeDrawable, PictureDrawable, LayerDrawable, και πολλών άλλων. Φυσικά, μπορούμε να επεκτείνουμε αυτές τις κλάσεις και να ορίσουμε δικά μας, προσαρμοσμένα αντικείμενα Drawable.

Υπάρχουν τρεις τρόποι για να ορίσουμε και να δημιουργήσουμε ένα Drawable: α) χρησιμοποιώντας μια εικόνα η οποία είναι αποθηκευμένη στους πόρους (resources) του πρότζεκτ, β) χρησιμοποιώντας ένα XML αρχείο το οποίο ορίζει τις ιδιότητες του Drawable, γ) χρησιμοποιώντας τους δομητές των κλάσεων.

Δημιουργώντας από εικόνες που βρίσκονται στους πόρους

Ένας απλός τρόπος να προσθέσουμε γραφικά σε μία εφαρμογή είναι να αναφερθούμε (reference) σε αρχεία εικόνων τα οποία βρίσκονται στους πόρους του πρότζεκτ. Οι υποστηριζόμενοι τύποι είναι PNG (προτείνεται), JPG (αποδεκτό) και GIF (δεν προτείνεται). Αυτή η τεχνική είναι προτιμητέα για εικονίδια εφαρμογών, λογότυπα, ή άλλα γραφικά όπως αυτά που χρησιμοποιούνται στα παιχνίδια.

Για να χρησιμοποιήσουμε μία εικόνα από τους πόρους, αρκεί απλώς να την τοποθετήσουμε στο φάκελο res/drawable/ του πρότζεκτ. Από εκεί, μπορούμε να

αναφερθούμε σε αυτήν από τον κώδικα ή από ένα αρχείο XML. Από όπου και αν αναφερθούμε θα πρέπει να χρησιμοποιήσουμε ένα ID πόρου (resource ID), το οποίο είναι το όνομα του αρχείου χωρίς την επέκταση (Π.χ., μπορούμε να αναφερθούμε στο αρχείο my_image.png με το my_image).

Δημιουργώντας με χρήση XML

Ο ορισμός ενός Drawable με XML αποτελεί καλή επιλογή, όταν αυτό δεν εξαρτάται αρχικά από μεταβλητές οι οποίες ορίζονται στον κώδικα της εφαρμογής ή όταν δεν απαιτείται αλληλεπίδραση από τον χρήστη για την δημιουργία του. Ακόμα και αν αναμένουμε ότι θα αλλάξουν οι ιδιότητες του Drawable, η χρήση XML αποτελεί μία επιλογή, εφόσον μπορούμε να τροποποιήσουμε τις ιδιότητές του αφού δημιουργηθεί.

Αφού ορίσουμε το Drawable σε XML, θα πρέπει να σώσουμε το XML αρχείο στο φάκελο res/drawable/ του πρότζεκτ. Ύστερα, μπορούμε να δημιουργήσουμε το αντικείμενο καλώντας την μέθοδο Resources.getDrawable(), περνώντας σαν παράμετρο το ID πόρου του XML αρχείου.

Όποιο αντικείμενο Drawable υποστηρίζει την μέθοδο inflate(), μπορεί να οριστεί σε XML και να δημιουργηθεί από την εφαρμογή. Κάθε Drawable που υποστηρίζει XML inflation χρησιμοποιεί συγκεκριμένες XML ιδιότητες με τις οποίες μπορούν να οριστούν οι ιδιότητες του αντικειμένου.

OpenGL ES

Το Android υποστηρίζει 2D και 3D γραφικά υψηλής απόδοσης με την Open Graphics Library, συγκεκριμένα, με την OpenGL ES. Η OpenGL αποτελεί ένα cross-platform API γραφικών, το οποίο καθορίζει μία διασύνδεση λογισμικού (software interface) για υλικό επεξεργασίας 3D γραφικών. Η OpenGL ES είναι μία έκδοση της προδιαγραφής OpenGL, σχεδιασμένη για ενσωματωμένες συσκευές. Το Android υποστηρίζει αρκετές εκδόσεις του OpenGL ES API:

- OpenGL ES 1.0 και 1.1 – Αυτή η προδιαγραφή API υποστηρίζεται από την έκδοση 1.0 του Android ή κάποια νεότερη.
- OpenGL ES 2.0 – Αυτή η προδιαγραφή API υποστηρίζεται από την έκδοση 2.2 του Android ή κάποια νεότερη.
- OpenGL ES 3.0 - Αυτή η προδιαγραφή API υποστηρίζεται από την έκδοση 4.3 του Android ή κάποια νεότερη.
- OpenGL ES 3.1 - Αυτή η προδιαγραφή API υποστηρίζεται από την έκδοση 5.0 του Android ή κάποια νεότερη.

Σημείωση: Μία συσκευή η οποία τρέχει Android 4.3 μπορεί να μην υποστηρίζει την προδιαγραφή OpenGL ES 3.0. Πέρα από την έκδοση του Android, απαιτείται μία υλοποίηση αυτής της graphics pipeline από τον κατασκευαστή της συσκευής.

Τα βασικά

Το Android υποστηρίζει την OpenGL μέσω του framework API αλλά και του Native Development Kit (NDK). Παρακάτω γίνεται μία αναφορά στις διασυνδέσεις του Android framework.

Υπάρχουν δύο βασικές κλάσεις στο framework του Android με τις οποίες μπορούμε να δημιουργήσουμε και να χειριστούμε γραφικά με το OpenGL ES API: GLSurfaceView και GLSurfaceView.Renderer.

GLSurfaceView

Αυτή η κλάση είναι ένα View στο οποίο μπορούμε να σχεδιάσουμε και να χειριστούμε αντικείμενα χρησιμοποιώντας κλήσεις (calls) του OpenGL API, και είναι παρόμοια στη λειτουργία με την SurfaceView. Μπορούμε να χρησιμοποιήσουμε αυτήν την κλάση δημιουργώντας ένα στιγμιότυπο της GLSurfaceView και προσθέτοντας τον Renderer σε αυτό. Ωστόσο, αν επιθυμούμε να συλλάβουμε (capture) συμβάντα της οθόνης αφής (touch screen events), θα πρέπει να επεκτείνουμε την κλάση GLSurfaceView και να υλοποιήσουμε τους ανιχνευτές αφής (touch listeners).

GLSurfaceView.Renderer

Αυτή η διασύνδεση ορίζει τις μεθόδους οι οποίες απαιτούνται για την σχεδίαση γραφικών σε ένα GLSurfaceView. Πρέπει να υλοποιηθεί σε μία ξεχωριστή κλάση και να συνδεθεί με το στιγμιότυπο του GLSurfaceView χρησιμοποιώντας την μέθοδο GLSurfaceView.setRenderer().

Για την διασύνδεση GLSurfaceView.Renderer θα πρέπει να υλοποιηθούν οι ακόλουθες μέθοδοι:

- onSurfaceCreated(): Το σύστημα καλεί αυτήν την μέθοδο μία φορά, όταν δημιουργεί το GLSurfaceView. Μπορούμε να χρησιμοποιήσουμε την μέθοδο για να ορίσουμε ενέργειες οι οποίες χρειάζεται να συμβούν μόνο μία φορά, όπως είναι η ρύθμιση των παραμέτρων περιβάλλοντος (environment parameters) ή η αρχικοποίηση γραφικών αντικειμένων OpenGL.
- onDrawFrame(): Το σύστημα καλεί αυτήν την μέθοδο σε κάθε επανασχεδίαση του GLSurfaceView. Αυτή η μέθοδος είναι το κύριο σημείο εκτέλεσης για την σχεδίαση (και την επανασχεδίαση) γραφικών αντικειμένων.
- onSurfaceChanged(): Το σύστημα καλεί αυτήν την μέθοδο όταν η γεωμετρία του GLSurfaceView αλλάζει, συμπεριλαμβανομένων αλλαγών στο μέγεθος του GLSurfaceView ή στον προσανατολισμό της συσκευής. Για παράδειγμα, αυτή η μέθοδος καλείται όταν η συσκευή αλλάξει από κατακόρυφο (portrait) σε οριζόντιο (landscape) προσανατολισμό.

Επιλέγοντας μία έκδοση του OpenGL API

Η έκδοση 1.0 του OpenGL ES API (και οι επεκτάσεις με την 1.1), η έκδοση 2.0, και η έκδοση 3.0, παρέχουν όλες διασυνδέσεις γραφικών υψηλής απόδοσης για την δημιουργία 3D παιχνιδιών, απεικονίσεων (visualizations) και διεπαφών χρήστη. Ο προγραμματισμός γραφικών για την OpenGL ES 2.0 και 3.0 είναι σε μεγάλο βαθμό παρόμοιος, με την έκδοση 3.0 να αποτελεί υπερσύνολο του 2.0 API διαθέτοντας επιπλέον χαρακτηριστικά. Από την άλλη ο προγραμματισμός για το OpenGL ES 1.0/1.1 API έναντι του OpenGL ES 2.0 και 3.0 διαφέρει σημαντικά, και έτσι οι προγραμματιστές θα πρέπει να λάβουν υπόψη τους παρακάτω παράγοντες πριν ξεκινήσουν την ανάπτυξη με αυτά τα APIs:

- **Απόδοση** – Σε γενικές γραμμές, η OpenGL ES 2.0 και 3.0 παρέχουν καλύτερη απόδοση γραφικών σε σχέση με τα ES 1.0/1.1 APIs. Ωστόσο, η διαφορά στην απόδοση μπορεί να ποικίλει ανάλογα με την συσκευή στην οποία τρέχει η εφαρμογή, λόγω διαφορών στην υλοποίηση της OpenGL ES graphics pipeline από τον κατασκευαστή.
- **Συμβατότητα με συσκευές** – Οι προγραμματιστές θα πρέπει να λάβουν υπόψη τους τους τύπους των συσκευών, τις εκδόσεις του Android, και τις εκδόσεις της OpenGL ES τις οποίες έχουν οι πελάτες τους. Πρέπει να σημειωθεί ότι το OpenGL ES 3.x API είναι προς τα πίσω συμβατό με το 2.0 API, το οποίο σημαίνει ότι μπορούμε να είμαστε πιο ευέλικτοι με την υλοποίηση της OpenGL ES στην εφαρμογή μας. Δηλώνοντας το OpenGL ES 2.0 API σαν προαπαιτούμενο στο AndroidManifest, μπορούμε να χρησιμοποιήσουμε αυτήν την έκδοση σαν βάση, να ελέγξουμε τη διαθεσιμότητα του 3.x API κατά τον χρόνο εκτέλεσης, και ύστερα να χρησιμοποιήσουμε χαρακτηριστικά της OpenGL ES 3.x αν η συσκευή τα υποστηρίζει.
- **Ευκολία προγραμματισμού** – Το OpenGL ES 1.0/1.1 API παρέχει «βολικές» συναρτήσεις (convenience functions) οι οποίες δεν υπάρχουν στα OpenGL ES 2.0 ή 3.0 APIs. Προγραμματιστές οι οποίοι είναι νέοι στην OpenGL ES μπορεί να βρουν ευκολότερο και γρηγορότερο τον προγραμματισμό στο 1.0/1.1 API.
- **Έλεγχος γραφικών** – Τα OpenGL ES 2.0 και 3.0 APIs παρέχουν μεγαλύτερο έλεγχο στον σχεδιασμό των γραφικών. Με πιο άμεσο έλεγχο της graphics pipeline, οι προγραμματιστές μπορούν να δημιουργήσουν εφέ τα οποία θα ήταν πολύ δύσκολο να δημιουργηθούν χρησιμοποιώντας το 1.0/1.1 API.
- **Υποστήριξη υφών (textures)** – Το OpenGL ES 3.0 API παρέχει την καλύτερη υποστήριξη για συμπίεση υφών γιατί εγγυάται την διαθεσιμότητα του φορμά συμπίεσης ETC2, το οποίο υποστηρίζει διάφανες υφές. Οι υλοποιήσεις των 1.x και 2.0 APIs συνήθως περιλαμβάνουν υποστήριξη για το ETC1, ωστόσο αυτό το φορμά δεν υποστηρίζει διάφανες υφές και έτσι θα πρέπει να παρέχουμε πόρους (resources) σε άλλα φορμά συμπίεσης τα οποία υποστηρίζονται από τις συσκευές στις οποίες στοχεύουμε.

Μολονότι η απόδοση, η συμβατότητα, η ευκολία, ο έλεγχος και άλλοι παράγοντες μπορεί να επηρεάσουν την απόφασή μας, θα πρέπει να

επιλέξουμε μία έκδοση του OpenGL API με βάση το ποια πιστεύουμε ότι παρέχει την καλύτερη εμπειρία για τους χρήστες.

Επιτάχυνση Υλικού

Από την έκδοση 3.0 του Android και έπειτα, η 2D rendering pipeline του Android υποστηρίζει επιτάχυνση υλικού, που σημαίνει ότι όλες οι λειτουργίες σχεδίασης που γίνονται στο Canvas ενός View χρησιμοποιούν τον επεξεργαστή γραφικών (GPU). Λόγω των αυξημένων πόρων που απαιτούνται για την ενεργοποίηση της επιτάχυνσης υλικού, η εφαρμογή θα καταναλώνει περισσότερη RAM.

Η επιτάχυνση υλικού είναι ενεργοποιημένη εξαρχής αν στοχεύουμε σε επίπεδο API >= 14, αλλά μπορεί να δηλωθεί και από τον προγραμματιστή. Αν η εφαρμογή μας χρησιμοποιεί μόνο Views και Drawables τα οποία παρέχονται από το Android, η καθολική ενεργοποίηση δεν θα πρέπει να προκαλέσει σχεδιαστικά προβλήματα. Ωστόσο, επειδή η επιτάχυνση υλικού δεν υποστηρίζεται από όλες τις 2D λειτουργίες σχεδίασης, η ενεργοποίησή της μπορεί να επηρεάσει κάποια από τα προσαρμοσμένα (custom) Views ή κάποιες από τις κλήσεις σχεδίασης. Τα προβλήματα συνήθως παρουσιάζονται με τη μορφή αόρατων στοιχείων, εξαιρέσεων, ή λανθασμένα τοποθετημένων pixels. Για να αντιμετωπιστεί αυτό το πρόβλημα, το Android μας δίνει την επιλογή να ενεργοποιήσουμε ή να απενεργοποιήσουμε την επιτάχυνση υλικού σε διάφορα επίπεδα.

Αν η εφαρμογή μας πραγματοποιεί προσαρμοσμένη σχεδίαση, θα πρέπει να την δοκιμάσουμε σε πραγματικές συσκευές με ενεργοποιημένη την επιτάχυνση υλικού για να εντοπίσουμε τυχόν προβλήματα.

Έλεγχος της επιτάχυνσης υλικού

Μπορούμε να ελέγξουμε την επιτάχυνση υλικού (ενεργοποιώντας ή απενεργοποιώντας την) στα ακόλουθα επίπεδα:

- Εφαρμογή
- Δραστηριότητα
- Παράθυρο (Στη παρούσα φάση μπορεί μόνο να ενεργοποιηθεί η επιτάχυνση υλικού σε επίπεδο παραθύρου, και όχι να απενεργοποιηθεί)
- View (Στη παρούσα φάση μπορεί μόνο να απενεργοποιηθεί η επιτάχυνση υλικού σε επίπεδο View, και όχι να ενεργοποιηθεί)

Τα μοντέλα σχεδίασης του Android

Όταν η επιτάχυνση υλικού είναι ενεργοποιημένη, το framework του Android χρησιμοποιεί ένα νέο μοντέλο σχεδίασης το οποίο χρησιμοποιεί λίστες εμφάνισης (display lists) για να αποδώσει (render) τα γραφικά της εφαρμογής στην οθόνη.

Μοντέλο σχεδίασης χωρίς επιτάχυνση υλικού (software-based)

Σε αυτό το μοντέλο σχεδίασης, τα Views σχεδιάζονται με τα ακόλουθα δύο βήματα:

1. Ακύρωση της ιεραρχίας
2. Σχεδίαση της ιεραρχίας

Όταν μία εφαρμογή χρειάζεται να ανανεώσει μέρος της διεπαφής χρήστη, καλεί την μέθοδο invalidate() (ή μία από τις παραλλαγές της) σε όποιο View έχει αλλάξει περιεχόμενο. Τα μηνύματα ακύρωσης μεταδίδονται μέχρι την κορυφή της ιεραρχίας για να υπολογίσουν τις περιοχές της οθόνης οι οποίες πρέπει να επανασχεδιαστούν (η «βρώμικη» περιοχή – dirty region). Στη συνέχεια, το Android σχεδιάζει όποιο View στην ιεραρχία τέμνεται με την βρώμικη περιοχή.

Μοντέλο σχεδίασης με επιτάχυνση υλικού

Το Android συνεχίζει να χρησιμοποιεί τις invalidate() και draw() για να ζητήσει να ανανεωθεί η οθόνη και για να αποδώσει Views, αλλά χειρίζεται την σχεδίαση διαφορετικά. Αντί να εκτελέσει τις εντολές σχεδίασης αμέσως, το σύστημα τις καταγράφει μέσα σε λίστες εμφάνισης, οι οποίες περιέχουν εξαγόμενο κώδικα από τον κώδικα σχεδίασης της ιεραρχίας των Views. Μία βελτιστοποίηση είναι ότι το Android χρειάζεται μόνο να καταγράψει και να ενημερώσει τις λίστες εμφάνισης για Views οι οποίες έχουν σημειωθεί σαν «βρώμικες» από μια κλήση invalidate(). Views τα οποία δεν έχουν ακυρωθεί μπορούν να επανασχεδιαστούν με τις εντολές οι οποίες βρίσκονται στην λίστα εμφάνισης η οποία είχε καταγραφεί προηγουμένως. Το νέο μοντέλο σχεδίασης περιλαμβάνει τρία στάδια:

1. Ακύρωση της ιεραρχίας
2. Καταγραφή και ενημέρωση των λιστών εμφάνισης
3. Σχεδίαση των λιστών εμφάνισης

Η χρήση λιστών εμφάνισης βελτιώνει την απόδοση των animations διότι όταν αλλάζουν οι τιμές για ιδιότητες, όπως άλφα ή περιστροφή, δεν χρειάζεται να ακυρωθεί το View. Αυτή η βελτιστοποίηση εφαρμόζεται επίσης σε Views με λίστες εμφάνισης (όλα τα Views όταν είναι ενεργοποιημένη η επιτάχυνση υλικού για την εφαρμογή). Για παράδειγμα, ας υποθέσουμε ότι έχουμε ένα LinearLayout το οποίο περιέχει μία ListView πάνω από ένα Button. Η λίστα εμφάνισης για το LinearLayout είναι η εξής:

- DrawDisplayList(ListView)
- DrawDisplayList(Button)

Ας υποθέσουμε τώρα ότι θέλουμε να αλλάξουμε το opacity της ListView. Αφού κληθεί η setAlpha(0.5f) για την ListView, η λίστα εμφάνισης περιέχει τώρα τα εξής:

- SaveLayerAlpha(0.5)
- DrawDisplayList(ListView)
- Restore

- DrawDisplayList(Button)

Ο περίπλοκος κώδικας σχεδίασης της ListView δεν εκτελέστηκε. Αντ' αυτού, το σύστημα ενημέρωσε μόνο την λίστα εμφάνισης του πολύ απλούστερου LinearLayout. Σε μία εφαρμογή όπου η επιτάχυνση υλικού είναι απενεργοποιημένη, ο κώδικας σχεδίασης της λίστας και του μητρικού στοιχείου (το LinearLayout σε αυτό το παράδειγμα) θα εκτελούνταν ξανά.

View Layers

Σε όλες τις εκδόσεις του Android, τα Views είχαν τη δυνατότητα να αποδοθούν σε buffers εκτός οθόνης (off-screen buffers), είτε χρησιμοποιώντας την cache σχεδίασης του View, είτε χρησιμοποιώντας Canvas.saveLayer(). Οι buffers εκτός οθόνης ή layers, έχουν διάφορες χρήσεις. Μπορούμε να τους χρησιμοποιήσουμε για να πάρουμε καλύτερη απόδοση όταν έχουμε animations με σύνθετα Views ή για να εφαρμόσουμε εφέ σύνθεσης. Για παράδειγμα, μπορούμε να υλοποιήσουμε εφέ σβησίματος (fade effects) χρησιμοποιώντας Canvas.saveLayer() για να αποδοθεί προσωρινά ένα View σε ένα layer και στη συνέχεια να το μεταφέρουμε στην οθόνη.

View layers και animations

Τα layers υλικού (hardware layers) παρέχουν γρηγορότερα και ομαλότερα animations όταν είναι ενεργοποιημένη η επιτάχυνση υλικού. Δεν είναι πάντα εφικτό να τρέχει ένα animation με 60 καρέ το δευτερόλεπτο όταν έχουμε να κάνουμε με animations σύνθετων Views. Μπορούμε να χρησιμοποιήσουμε layers υλικού για να αποδοθεί το View σε ένα hardware texture. Το hardware texture μπορεί ύστερα να χρησιμοποιηθεί για το animation του View, χωρίς έτσι να υπάρχει η ανάγκη συνεχούς επανασχεδίασης του View κατά τη διάρκεια του animation. Το View επανασχεδιάζεται μόνο αν αλλάξουν οι ιδιότητές του, οπότε και καλείται η invalidate(), ή αν καλέσουμε εμείς οι ίδιοι την invalidate().

Λόγω του ότι τα layers υλικού καταναλώνουν μνήμη βίντεο, προτείνεται να τα έχουμε ενεργοποιημένα για όσο κρατάει το animation και να τα απενεργοποιούμε με την ολοκλήρωση του animation.

	Hardware layer	DisplayList	Software
Time in ms	0.009	2.1	10.3

Μετρήσεις από την σχεδίαση μιας ListView με Android 3.0 σε ένα Motorola XOOM

Animation

Το framework του Android παρέχει δύο συστήματα animation: property animation (εισήχθη με το Android 3.0) και view animation. Γενικά, το σύστημα property animation είναι το προτιμότερο, επειδή είναι πιο ευέλικτο και προσφέρει περισσότερα χαρακτηριστικά. Πέρα από αυτά τα δύο συστήματα, μπορούμε να χρησιμοποιήσουμε το Drawable animation, το οποίο μας επιτρέπει να φορτώσουμε μία σειρά από Drawables από τους πόρους (resources) και να τα εμφανίσουμε το ένα μετά το άλλο.

Property Animation

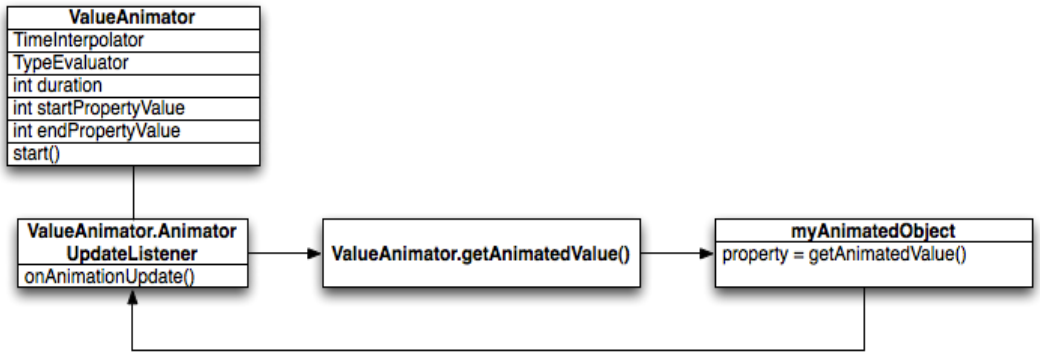
Το σύστημα property animation είναι ένα ισχυρό framework το οποίο μας επιτρέπει να φτιάχνουμε animations με σχεδόν οτιδήποτε. Μπορούμε να ορίσουμε ένα animation για να αλλάξουμε οποιαδήποτε ιδιότητα ενός αντικειμένου για κάποιο χρονικό διάστημα, είτε αυτό σχεδιάζεται στην οθόνη είτε όχι. Ένα property animation αλλάζει την τιμή μιας ιδιότητας (ένα πεδίο σε ένα αντικείμενο) για κάποιο προσδιορισμένο χρονικό διάστημα. Για να φτιάξουμε ένα animation με κάποιο αντικείμενο, προσδιορίζουμε την ιδιότητα της οποίας η τιμή θέλουμε να αλλάζει, όπως είναι η θέση ενός αντικειμένου στην οθόνη, το χρονικό διάστημα μέσα στο οποίο θα αλλάζει αυτή η τιμή, και τις τιμές μεταξύ των οποίων θέλουμε να κινηθεί η τιμή της ιδιότητας.

Το σύστημα property animation μας επιτρέπει να ορίσουμε τα ακόλουθα χαρακτηριστικά ενός animation:

- Διάρκεια: Μπορούμε να ορίσουμε την διάρκεια ενός animation. Η προεπιλογή είναι 300 ms.
- Παρεμβολή χρόνου (Time interpolation): Μπορούμε να ορίσουμε τον τρόπο με τον οποίο οι τιμές της ιδιότητας υπολογίζονται σαν συνάρτηση του τρέχοντα χρόνου που έχει περάσει.
- Αριθμός επαναλήψεων και συμπεριφορά: Μπορούμε να ορίσουμε το εάν θα επαναληφθεί το animation όταν τελειώσει, καθώς και το πόσες φορές θα επαναληφθεί. Μπορούμε επίσης να ορίσουμε το εάν θέλουμε να παίξει αντίστροφα το animation (reverse). Αν το βάλουμε στο reverse, το animation παίζει μία φορά κανονικά προς τα μπροστά, και μετά προς τα πίσω επανειλημμένα, έως ότου συμπληρωθεί ο αριθμός των επαναλήψεων.
- Animator sets: Μπορούμε να ομαδοποιήσουμε τα animations σε λογικά σετ τα οποία παίζουν ταυτόχρονα ή διαδοχικά ή ύστερα από προκαθορισμένες καθυστερήσεις.
- Καθυστέρηση ανανέωσης καρέ: Μπορούμε να ορίσουμε το πόσο συχνά θα ανανεώνονται τα καρέ του animation. Η προεπιλογή είναι να ανανεώνονται κάθε 10 ms, αλλά η ταχύτητα με την οποία μπορεί μία εφαρμογή να ανανεώνει τα καρέ εξαρτάται τελικά από το πόσο απασχολημένο είναι το σύστημα και από το πόσο γρήγορα μπορεί το σύστημα να εξυπηρετήσει τον χρονιστή.

Πώς δουλεύει το Property Animation

Η ακόλουθη εικόνα απεικονίζει το πώς δουλεύουν οι βασικές κλάσεις μεταξύ τους.



Πώς γίνονται οι υπολογισμοί στα animations

Το αντικείμενο ValueAnimator κρατάει πληροφορίες για τον χρονισμό του animation, όπως είναι η τρέχουσα τιμή της ιδιότητας την οποία κάνει animate, και το πόση ώρα τρέχει το animation.

Ο ValueAnimator ενθυλακώνει έναν TimeInterpolator, ο οποίος ορίζει την παρεμβολή (interpolation) του animation, και έναν TypeEvaluator, ο οποίος ορίζει το πώς υπολογίζονται οι τιμές της ιδιότητας.

Για να ξεκινήσουμε ένα animation, μπορούμε να δημιουργήσουμε έναν ValueAnimator και να του δώσουμε την αρχική και την τελική τιμή της ιδιότητας που θέλουμε να κάνουμε animate, όπως επίσης και την διάρκεια του animation. Όταν καλούμε την start() το animation ξεκινάει. Κατά την διάρκεια του animation, ο ValueAnimator υπολογίζει ένα κλάσμα του χρόνου που έχει περάσει (elapsed fraction), ανάμεσα στο 0 και το 1, με βάση την διάρκεια του animation, και τον χρόνο που έχει περάσει. Το elapsed fraction, αναπαριστά το ποσοστό του χρόνου που έχει περάσει, 0 σημαίνει 0% και 1 σημαίνει 100%.

Όταν ο ValueAnimator υπολογίσει το elapsed fraction, καλεί τον TimeInterpolator για να υπολογίσει το λεγόμενο interpolated fraction. Το interpolated fraction αντιστοιχίζει το elapsed fraction σε ένα νέο κλάσμα, το οποίο παίρνει υπόψη του τον TimeInterpolator που χρησιμοποιείται στο animation. Ένας τύπος TimeInterpolator είναι ο LinearInterpolator, με τον οποίο ο ρυθμός αλλαγής είναι σταθερός. Ο AccelerateDecelerateInterpolator είναι ένας άλλος TimeInterpolator με τον οποίο ο ρυθμός αλλαγής ξεκινάει και τελειώνει αργά, αλλά επιταχύνει στο ενδιάμεσο.

Αφού υπολογιστεί το interpolated fraction, ο ValueAnimator καλεί τον κατάλληλο TypeEvaluator, για να υπολογίσει την τιμή της ιδιότητας, με βάση το interpolated fraction, την αρχική τιμή, και την τελική τιμή του animation.

Που διαφέρει το Property Animation από το View Animation

Το σύστημα view animation μας δίνει τη δυνατότητα να δημιουργήσουμε animations μόνο με αντικείμενα View, οπότε αν θέλαμε να φτιάξουμε animations με άλλα αντικείμενα, θα έπρεπε να υλοποιήσουμε δικό μας κώδικα για αυτόν τον σκοπό. Το σύστημα view animation περιορίζεται επίσης από το γεγονός ότι μπορεί να κάνει animate μόνο ορισμένα από τα χαρακτηριστικά ενός View, όπως την κλίμακα (scaling) και την περιστροφή, αλλά όχι το χρώμα φόντου, για παράδειγμα.

Ένα άλλο μειονέκτημα του συστήματος view animation είναι ότι τροποποιεί την θέση στην οποία σχεδιάζεται το View, και όχι το ίδιο το αντικείμενο. Για παράδειγμα, αν κάνουμε animate ένα κουμπί έτσι ώστε να μετακινηθεί κατά μήκος της οθόνης, το κουμπί σχεδιάζεται σωστά, αλλά η περιοχή στην οποία μπορούμε να κάνουμε κλικ δεν αλλάζει, οπότε πρέπει να υλοποιήσουμε τη λογική για να γίνει αυτό.

Το σύστημα property animation δεν έχει αυτούς τους περιορισμούς. Μπορούμε να κάνουμε animate όλες τις ιδιότητες ενός αντικειμένου, και επιπλέον τροποποιείται το ίδιο το αντικείμενο.

Παρ' όλα αυτά, με το view animation χρειάζεται να γράψουμε λιγότερο κώδικα. Αν με το view animation έχουμε το επιθυμητό αποτέλεσμα, δεν είναι απαραίτητο να χρησιμοποιήσουμε property animation.

Φτιάχνοντας animations με τον ValueAnimator

Η κλάση ValueAnimator μας επιτρέπει να κάνουμε animate τιμές κάποιου τύπου ορίζοντας τις τιμές μεταξύ των οποίων θα κινηθεί η τιμή της ιδιότητας. Μπορούμε να πάρουμε έναν ValueAnimator χρησιμοποιώντας μία από τις μεθόδους: ofInt(), ofFloat(), ή ofObject(). Για παράδειγμα:

```
ValueAnimator animation = ValueAnimator.ofFloat(0f, 1f);
animation.setDuration(1000);
animation.start();
```

Σε αυτόν τον κώδικα, ο ValueAnimator ξεκινάει τους υπολογισμούς για τις τιμές του animation, οι οποίες κινούνται μεταξύ 0 και 1, για ένα διάστημα 1000 ms, μετά την κλήση της μεθόδου start().

Μπορούμε επίσης να κάνουμε animate έναν προσαρμοσμένο τύπο (custom type) ως εξής:

```
ValueAnimator animation = ValueAnimator.ofObject(new MyTypeEvaluator(),
startPropertyValue, endPropertyValue);
animation.setDuration(1000);
animation.start()
```

Ωστόσο, τα προηγούμενα τμήματα κώδικα δεν επηρεάζουν κάποιο αντικείμενο, διότι ο ValueAnimator δεν τροποποιεί απευθείας αντικείμενα ή ιδιότητες. Για να

τροποποιήσουμε τα αντικείμενα με τις υπολογισθείσες τιμές, θα πρέπει να ορίσουμε ανιχνευτές (listeners) στον ValueAnimator για να χειριστούμε σημαντικά γεγονότα, όπως είναι οι ανανεώσεις των καρέ (frame updates). Όταν υλοποιούμε τους ανιχνευτές, μπορούμε να πάρουμε την υπολογισθείσα τιμή για ένα συγκεκριμένο καρέ καλώντας την getAnimatedValue().

Φτιάχνοντας animations με τον ObjectAnimator

Ο ObjectAnimator είναι μία υποκλάση του ValueAnimator. Συνδυάζει την μηχανή χρονισμού (timing engine) και τον υπολογισμό των τιμών του ValueAnimator με την ικανότητα να κάνει animate κάποια ιδιότητα ενός αντικειμένου. Κατ' αυτό τον τρόπο, μπορούμε να κάνουμε πολύ πιο εύκολα animate κάποιο αντικείμενο. Δεν χρειάζεται πλέον να υλοποιήσουμε τον ValueAnimator.AnimatorUpdateListener, λόγω του ότι οι τιμές των ιδιοτήτων του αντικειμένου τροποποιούνται αυτόματα.

Ο κώδικας για την κατασκευή ενός ObjectAnimator είναι παρόμοιος με τον κώδικα για τον ValueAnimator. Η διαφορά είναι ότι θα πρέπει να προσθέσουμε το αντικείμενο και το όνομα της ιδιότητας:

```
ObjectAnimator anim = ObjectAnimator.ofFloat(foo, "alpha", 0f, 1f);
anim.setDuration(1000);
anim.start();
```

Δηλώνοντας τα animations σε XML

Το σύστημα property animation μας επιτρέπει να ορίσουμε τα animations με XML, αντί να το κάνουμε προγραμματιστικά. Ορίζοντας τα animations σε XML, μπορούμε να τα χρησιμοποιήσουμε εύκολα σε πολλές Δραστηριότητες και να επεξεργαστούμε πιο εύκολα την σειρά εκτέλεσής τους.

Οι ακόλουθες κλάσεις για property animation υποστηρίζουν XML με τις ακόλουθες ετικέτες XML:

- ValueAnimator - <animator>
- ObjectAnimator - <objectAnimator>
- AnimatorSet - <set>

Στο ακόλουθο παράδειγμα, υπάρχουν δύο animations τα οποία παίζουν ταυτόχρονα και στη συνέχεια ακολουθεί ένα τρίτο. Με τα δύο πρώτα animations αλλάζει η θέση του αντικειμένου στην οθόνη, ενώ το τρίτο κάνει το αντικείμενο διάφανο:

```
<set android:ordering="sequentially">
    <set>
        <objectAnimator
            android:propertyName="x"
            android:duration="500"
            android:valueTo="400"
            android:valueType="intType"/>
        <objectAnimator
```

```
            android:propertyName="y"
            android:duration="500"
            android:valueTo="300"
            android:valueType="intType"/>
    </set>
    <objectAnimator
        android:propertyName="alpha"
        android:duration="500"
        android:valueTo="1f"/>

</set
```

Προκειμένου να τρέξουμε αυτό το animation, θα πρέπει να δημιουργήσουμε ένα αντικείμενο AnimatorSet δίνοντας σαν παράμετρο στην σχετική συνάρτηση την τοποθεσία του XML αρχείου (το τοποθετούμε στο φάκελο res/anim), και στη συνέχεια να ορίσουμε το αντικείμενο-στόχο του animation.

```
AnimatorSet set = (AnimatorSet) AnimatorInflater.loadAnimator(myContext,
    R.anim.property_animator);
set.setTarget(myObject);
set.start();
```

Drawable animation

Με το Drawable animation μπορούμε να φορτώσουμε μία σειρά από πόρους Drawable (Drawable resources) τον ένα μετά τον άλλο για να δημιουργήσουμε ένα animation. Αυτό είναι ένα παραδοσιακό animation με την έννοια ότι δημιουργείται με μία αλληλουχία διαφορετικών εικόνων, όπως συμβαίνει με ένα φιλμ. Η κλάση AnimationDrawable αποτελεί τη βάση για τη δημιουργία Drawable animations.

Μολονότι μπορούμε να ορίσουμε τα καρέ του animation προγραμματιστικά, χρησιμοποιώντας το AnimationDrawable API, αυτό επιτυγχάνεται πιο εύκολα με ένα αρχείο XML το οποίο περιλαμβάνει μία λίστα από καρέ τα οποία συνθέτουν το animation. Για αυτόν τον τύπο animation, το αρχείο XML θα πρέπει να τοποθετηθεί στα φάκελο res/drawable/ του πρότζεκτ. Σε αυτήν την περίπτωση, οι εντολές είναι η σειρά και η διάρκεια για το κάθε καρέ του animation.

Το αρχείο XML αποτελείται από το ριζικό στοιχείο <animation-list> και μία σειρά από θυγατρικούς κόμβους <item> ο καθένας από τους οποίους ορίζει ένα καρέ: ένα πόρο Drawable για το καρέ και την διάρκεια του καρέ. Το αρχείο αυτό μπορούμε στη συνέχεια να το θέσουμε σαν φόντο σε κάποιο View (π.χ. ImageView) και στη συνέχεια να το παίξουμε.

Ασφάλεια

Αρχιτεκτονική ασφάλειας

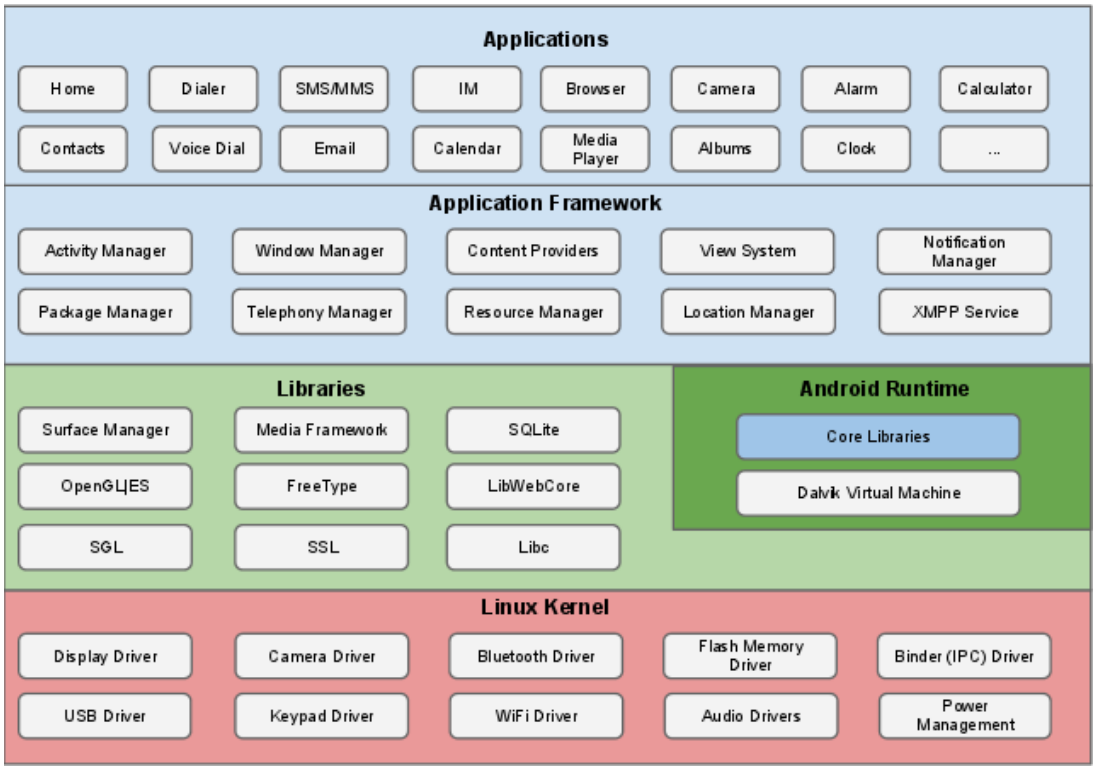
Το Android επιδιώκει να είναι το πιο λειτουργικό και ασφαλές λειτουργικό σύστημα για φορητές συσκευές επαναπροσδιορίζοντας παραδοσιακές μεθόδους ασφάλειας για λειτουργικά συστήματα προκειμένου:

- Να προστατέψει τα δεδομένα των χρηστών
- Να προστατέψει πόρους του συστήματος (συμπεριλαμβανομένου του δικτύου)
- Να παρέχει απομόνωση εφαρμογών

Για την επίτευξη αυτών των στόχων, το Android παρέχει τα εξής χαρακτηριστικά ασφάλειας:

- Ισχυρή ασφάλεια σε επίπεδο λειτουργικού μέσω του πυρήνα Linux
- Υποχρεωτικό φίλτρο (sandbox) για όλες τις εφαρμογές
- Ασφαλής επικοινωνία μεταξύ διεργασιών
- Υπογεγραμμένες εφαρμογές (application signing)
- Καθορισμός δικαιωμάτων από τις εφαρμογές και έγκριση αυτών από τους χρήστες

Στην ακόλουθη εικόνα φαίνεται η αρχιτεκτονική του Android. Κάθε στοιχείο υποθέτει ότι τα στοιχεία που βρίσκονται χαμηλότερα είναι σωστά ασφαλισμένα. Αν εξαιρέσουμε ένα μικρό τμήμα κώδικα του λειτουργικού το οποίο τρέχει σαν υπερχρήστης (root), όλος ο κώδικας που βρίσκεται πάνω από το πυρήνα του Linux περιορίζεται από το φίλτρο εφαρμογών (Application Sandbox).



Ασφάλεια σε επίπεδο συστήματος και πυρήνα

Σε επίπεδο λειτουργικού, το Android παρέχει την ασφάλεια του πυρήνα Linux, όπως επίσης ασφαλείς μηχανισμούς για την επικοινωνία μεταξύ διεργασιών προκειμένου να επικοινωνούν με ασφάλεια οι εφαρμογές οι οποίες τρέχουν σε διαφορετικές διεργασίες. Αυτά τα χαρακτηριστικά ασφάλειας σε επίπεδο λειτουργικού διασφαλίζουν ότι ακόμα και ο εγγενής κώδικας περιορίζεται από το φίλτρο εφαρμογών. Ανεξάρτητα από το αν αυτός ο κώδικας είναι αποτέλεσμα της συμπεριφοράς μιας εφαρμογής ή αν προκύπτει από την εκμετάλλευση κάποιας αδυναμίας μιας εφαρμογής, το σύστημα θα εμποδίσει την εφαρμογή από το να βλάψει άλλες εφαρμογές, το σύστημα, ή την ίδια την συσκευή.

Ασφάλεια Linux

Το θεμέλιο της πλατφόρμας Android είναι ο πυρήνας Linux. Η χρήση του είναι διαδεδομένη εδώ και χρόνια, ενώ χρησιμοποιείται σε εκατομμύρια συστήματα τα οποία θεωρούνται ευαίσθητα από πλευράς ασφάλειας. Μέσα από την έρευνα, τις επιθέσεις, και τις επιδιορθώσεις που έχουν γίνει από χιλιάδες προγραμματιστές, το Linux αποτελεί έναν σταθερό και ασφαλή πυρήνα τον οποίο εμπιστεύονται πολλές επιχειρήσεις και ειδικοί σε θέματα ασφάλειας.

Αποτελώντας την βάση ενός περιβάλλοντος για φορητές συσκευές, ο πυρήνας Linux παρέχει στο Android αρκετά σημαντικά χαρακτηριστικά ασφάλειας, όπως:

- Μοντέλο δικαιωμάτων το οποίο βασίζεται στον χρήστη
- Απομόνωση διεργασιών
- Ασφαλής επικοινωνία μεταξύ των διεργασιών
- Τη δυνατότητα αφαίρεσης περιττών τμημάτων του πυρήνα και τμημάτων τα οποία ενδεχομένως να μην είναι ασφαλή

Καθότι πολυχρηστικό (multiuser) λειτουργικό, βασικό ζητούμενο για το Linux είναι η απομόνωση των πόρων που χρησιμοποιούν οι χρήστες για λόγους ασφάλειας. Έτσι το Linux:

- Δεν επιτρέπει στο χρήστη Α να διαβάσει τα αρχεία του χρήστη Β
- Διασφαλίζει ότι ο χρήστης Α δεν εξαντλεί την μνήμη του χρήστη Β
- Διασφαλίζει ότι ο χρήστης Α δεν εξαντλεί τους πόρους επεξεργαστή του χρήστη Β
- Διασφαλίζει ότι ο χρήστης Α δεν εξαντλεί τις συσκευές του χρήστη Β (π.χ. τηλεφωνία, GPS, bluetooth)

Το φίλτρο εφαρμογών (Application Sandbox)

Η πλατφόρμα Android εκμεταλλεύεται το μοντέλο ασφάλειας του Linux προκειμένου να ταυτοποιήσει και να απομονώσει τους πόρους των εφαρμογών. Το Android αναθέτει ένα μοναδικό αναγνωριστικό χρήστη (UID) σε κάθε εφαρμογή Android και την τρέχει ως χρήστη σε μία ξεχωριστή διεργασία. Αυτή η προσέγγιση είναι διαφορετική από αυτήν που ακολουθούν άλλα

λειτουργικά συστήματα (συμπεριλαμβανομένου του Linux), όπου πολλές εφαρμογές τρέχουν με τα ίδια δικαιώματα χρήστη.

Με αυτόν τον τρόπο έχουμε ένα φίλτρο εφαρμογών σε επίπεδο πυρήνα. Ο πυρήνας επιβάλλει κανόνες ασφάλειας μεταξύ των εφαρμογών και του συστήματος σε επίπεδο διεργασίας χρησιμοποιώντας μηχανισμούς του Linux, όπως είναι τα αναγνωριστικά χρήστη και ομάδας τα οποία ανατίθενται σε εφαρμογές. Η προεπιλογή είναι να μην μπορούν οι εφαρμογές να αλληλεπιδρούν μεταξύ τους, ενώ επιπλέον οι εφαρμογές έχουν περιορισμένη πρόσβαση στο λειτουργικό σύστημα. Αν η εφαρμογή Α προσπαθήσει να κάνει κάτι κακόβουλο όπως είναι η ανάγνωση των δεδομένων της εφαρμογής Β ή η πραγματοποίηση κλήσης (υπάρχει ξεχωριστή εφαρμογή για πραγματοποίηση κλήσεων) χωρίς να έχει την άδεια, το λειτουργικό σύστημα αποτρέπει μία τέτοια ενέργεια επειδή η εφαρμογή Α δεν έχει τα απαραίτητα δικαιώματα χρήστη. Το φίλτρο είναι απλό, και βασίζεται στον τρόπο που διαχωρίζει το UNIX τις διεργασίες και τα δικαιώματα αρχείων διαφορετικών χρηστών.

Εφόσον το φίλτρο εφαρμογών είναι στον πυρήνα, αυτό το μοντέλο ασφάλειας εφαρμόζεται και για τον εγγενή κώδικα και για τις εφαρμογές που είναι προεγκατεστημένες. Όλο το λογισμικό που βρίσκεται πάνω από τον πυρήνα, στο οποίο συμπεριλαμβάνονται οι βιβλιοθήκες του λειτουργικού, το framework εφαρμογών, το περιβάλλον εκτέλεσης, και όλες οι εφαρμογές, τρέχει εντός του φίλτρου εφαρμογών. Σε ορισμένες πλατφόρμες, οι προγραμματιστές περιορίζονται σε ένα συγκεκριμένο framework, ή γλώσσα προκειμένου να επιβάλλουν κανόνες ασφάλειας. Στο Android, δεν υπάρχουν περιορισμοί αναφορικά με το πώς πρέπει να είναι γραμμένη μία εφαρμογή προκειμένου να επιβάλλονται κανόνες ασφάλειας. Ο εγγενής κώδικας είναι εξίσου ασφαλής με τον ερμηνευμένο (interpreted) κώδικα.

Σε ορισμένα λειτουργικά συστήματα, τα σφάλματα μνήμης (memory corruption errors) οδηγούν γενικά σε πλήρη παραβίαση της ασφάλειας της συσκευής. Αυτό δεν ισχύει για το Android, λόγω του ότι το φίλτρο εφαρμόζεται σε επίπεδο λειτουργικού για όλες τις εφαρμογές και τους πόρους τους. Κάποιο σφάλμα μνήμης θα οδηγήσει στην εκτέλεση αυθαίρετου κώδικα στο πλαίσιο της εφαρμογής στην οποία προκλήθηκε, με τα δικαιώματα τα οποία έχουν παραχωρηθεί από το λειτουργικό σύστημα.

Όπως όλοι οι μηχανισμοί ασφάλειας, το φίλτρο εφαρμογών δεν είναι απαραβίαστο. Ωστόσο, για να παρακάμψει κάποιος το φίλτρο εφαρμογών σε μία σωστά ρυθμισμένη συσκευή, θα πρέπει να παραβιάσει την ασφάλεια του πυρήνα Linux.

Διαμέρισμα Συστήματος (System Partition) και Ασφαλής Λειτουργία

Το διαμέρισμα συστήματος περιέχει τον πυρήνα του Android, τις βιβλιοθήκες του λειτουργικού συστήματος, το περιβάλλον εκτέλεσης μιας εφαρμογής, το framework εφαρμογών, και κάποιες εφαρμογές. Αυτό το διαμέρισμα είναι ρυθμισμένο έτσι ώστε να μπορεί μόνο να διαβασθεί. Όταν ένας χρήστης εκκινεί την συσκευή σε ασφαλή λειτουργία, μόνο οι προεγκατεστημένες εφαρμογές

είναι διαθέσιμες. Έτσι διασφαλίζεται ότι ο χρήστης μπορεί να ξεκινήσει το κινητό του σε ένα περιβάλλον στο οποίο δεν υπάρχουν εφαρμογές τρίτων.

Δικαιώματα συστήματος αρχείων

Σε ένα περιβάλλον το οποίο λειτουργεί όπως το UNIX, τα δικαιώματα του συστήματος αρχείων διασφαλίζουν ότι ένας χρήστης δεν μπορεί να τροποποιήσει ή να διαβάσει τα αρχεία κάποιου άλλου χρήστη. Στο Android, κάθε εφαρμογή τρέχει σαν ξεχωριστός χρήστης. Τα αρχεία τα οποία δημιουργούνται από μία εφαρμογή δεν μπορούν να διαβαστούν ή να τροποποιηθούν από άλλη εφαρμογή, εκτός και αν το ορίσει έτσι ο προγραμματιστής.

Κρυπτογράφηση

Το Android παρέχει για κρυπτογράφηση μία σειρά από APIs τα οποία μπορούν να χρησιμοποιηθούν από τις εφαρμογές. Σε αυτά περιλαμβάνονται υλοποιήσεις συχνά χρησιμοποιούμενων κρυπτογραφικών μεθόδων, όπως είναι η AES, RSA, DSA, και η SHA. Επιπλέον, το Android παρέχει APIs για πρωτόκολλα υψηλότερου επιπέδου όπως είναι το SSL και το HTTPS.

Συσκευές με δικαιώματα υπερχρήστη (rooting)

Από προεπιλογή, στο Android μόνο ο πυρήνας και ένα μικρό υποσύνολο των προεγκατεστημένων εφαρμογών τρέχουν με δικαιώματα υπερχρήστη. Το Android δεν απαγορεύει σε έναν χρήστη ή μία εφαρμογή με δικαιώματα υπερχρήστη να τροποποιήσουν το λειτουργικό σύστημα, τον πυρήνα, ή κάποια άλλη εφαρμογή. Γενικά, ο υπερχρήστης έχει πλήρη πρόσβαση σε όλες τις εφαρμογές και σε όλα τα δεδομένα των εφαρμογών. Οι χρήστες οι οποίοι αλλάζουν τα δικαιώματα σε μία συσκευή Android προκειμένου να παραχωρήσουν δικαιώματα υπερχρήστη σε μία εφαρμογή αυξάνουν την έκθεση ασφάλειας σε κακόβουλες εφαρμογές και σε πιθανά σφάλματα των εφαρμογών.

Για τους προγραμματιστές που εργάζονται με το Android είναι σημαντικό να μπορούν να τροποποιήσουν την συσκευή Android, την οποία έχουν. Σε πολλές συσκευές Android οι χρήστες έχουν την δυνατότητα να ξεκλειδώσουν το λογισμικό εκκίνησης του λειτουργικού συστήματος (bootloader) προκειμένου να μπορούν να εγκαταστήσουν κάποιο εναλλακτικό λειτουργικό σύστημα. Αυτά τα εναλλακτικά λειτουργικά συστήματα μπορεί να δίνουν τη δυνατότητα στον κάτοχο να αποκτήσει δικαιώματα υπερχρήστη, προκειμένου να μπορεί να εντοπίσει σφάλματα (debugging) σε εφαρμογές και στοιχεία του συστήματος ή να έχει πρόσβαση σε χαρακτηριστικά στα οποία οι εφαρμογές δεν έχουν πρόσβαση μέσω των Android APIs.

Σε ορισμένες συσκευές, κάποιος ο οποίος έχει φυσική πρόσβαση στην συσκευή και ένα USB καλώδιο, είναι σε θέση να εγκαταστήσει ένα νέο λειτουργικό σύστημα το οποίο του δίνει δικαιώματα υπερχρήστη. Προκειμένου να προστατέψει τα δεδομένα χρήστη τα οποία μπορεί να υπάρχουν στην συσκευή, ο μηχανισμός με τον οποίο ξεκλειδώνει ο bootloader, επιβάλλει την

διαγραφή των δεδομένων χρήστη σαν μέρος της διαδικασίας ξεκλειδώματος. Αν κάποιος καταφέρει να αποκτήσει δικαιώματα υπερχρήστη εκμεταλλευόμενος κάποιο σφάλμα στον πυρήνα ή κάποιο κενό ασφάλειας, μπορεί να παρακάμψει αυτήν την προστασία.

Η κρυπτογράφηση των δεδομένων των εφαρμογών με κάποιο κλειδί το οποίο είναι αποθηκευμένο στην συσκευή, δεν παρέχει προστασία από τους υπερχρήστες. Οι εφαρμογές μπορούν να προσθέσουν ένα επίπεδο προστασίας των δεδομένων κρυπτογραφώντας τα δεδομένα με ένα κλειδί το οποίο είναι αποθηκευμένο εκτός της συσκευής, όπως σε ένα διακομιστή, ή με ένα κωδικό χρήστη. Αυτή η προσέγγιση παρέχει προσωρινή προστασία, για όσο διάστημα το κλειδί δεν βρίσκεται στην συσκευή, αλλά σε κάποιο σημείο το κλειδί πρέπει να δοθεί στην εφαρμογή και τότε είναι διαθέσιμο πλέον σε όσους έχουν δικαιώματα υπερχρήστη.

Μία πιο αποτελεσματική προσέγγιση για την προστασία των δεδομένων από υπερχρήστες είναι οι λύσεις που αξιοποιούν το υλικό (hardware), όπως για παράδειγμα η προστασία πολυμέσων με DRM.

Το Android παρέχει επίσης τη δυνατότητα πλήρους κρυπτογράφησης του συστήματος αρχείων, η οποία χρησιμοποιεί τον κωδικό της συσκευής για να προστατέψει το κλειδί κρυπτογράφησης, οπότε σε περίπτωση κλοπής της συσκευής, η τροποποίηση του bootloader ή του λειτουργικού δεν αρκεί για να αποκτήσει ο υπερχρήστης πρόσβαση στα δεδομένα.

Ασφάλεια εφαρμογών στο Android

Το μοντέλο εκχώρησης δικαιωμάτων του Android: Η πρόσβαση σε προστατευμένα APIs

Όλες οι εφαρμογές στο Android περιορίζονται από το φίλτρο εφαρμογών, για το οποίο έγινε αναφορά προηγουμένως. Από προεπιλογή, μία εφαρμογή Android έχει περιορισμένη πρόσβαση στους πόρους του συστήματος. Το σύστημα διαχειρίζεται την πρόσβαση μιας εφαρμογής σε πόρους, οι οποίοι αν χρησιμοποιηθούν με λάθος τρόπο ή και κακόβουλα, θα μπορούσαν να επηρεάσουν αρνητικά την εμπειρία χρήστη, το δίκτυο, ή τα δεδομένα της συσκευής.

Αυτοί οι περιορισμοί υλοποιούνται με διάφορους τρόπους. Ορισμένες δυνατότητες περιορίζονται λόγω της εσκεμμένης έλλειψης APIs για συγκεκριμένες λειτουργίες. Για παράδειγμα, δεν υπάρχει API για τον απευθείας χειρισμό της κάρτας SIM. Σε κάποιες περιπτώσεις, ο διαχωρισμός των ρόλων είναι ένα μέτρο ασφάλειας, όπως με την απομόνωση των δεδομένων των εφαρμογών. Σε άλλες περιπτώσεις, τα ευαίσθητα APIs προορίζονται για χρήση μόνο από τις αξιόπιστες εφαρμογές και προστατεύονται από έναν μηχανισμό ασφάλειας που είναι γνωστός σαν Δικαιώματα:

Στα προστατευμένα APIs περιλαμβάνονται:

- Λειτουργίες κάμερας

- Δεδομένα για την τοποθεσία (GPS)
- Λειτουργίες Bluetooth
- Λειτουργίες τηλεφώνου
- Λειτουργίες SMS/MMS
- Συνδέσεις δικτύου/δεδομένων

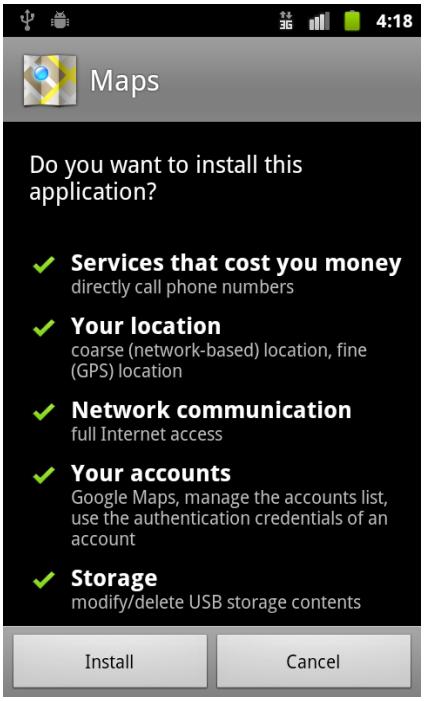
Αυτοί οι πόροι είναι προσβάσιμοι μόνο μέσω του λειτουργικού συστήματος. Για να χρησιμοποιήσει μία εφαρμογή τα προστατευμένα APIs, θα πρέπει να ορίσει τις δυνατότητες που χρειάζεται στο μανιφέστο της. Όταν προετοιμάζεται να εγκαταστήσει μία εφαρμογή, το σύστημα εμφανίζει ένα παράθυρο διαλόγου στο οποίο αναφέρονται τα δικαιώματα για τα οποία ζητείται άδεια από τον χρήστη, και το οποίο ρωτάει αν πρέπει να συνεχιστεί η εγκατάσταση. Αν ο χρήστης επιλέξει να συνεχίσει με την εγκατάσταση, το σύστημα θεωρεί ότι ο χρήστης έχει χορηγήσει όλα τα δικαιώματα που ζήτησε η εφαρμογή. Ο χρήστης δεν έχει τη δυνατότητα να εκχωρήσει ή να αρνηθεί μεμονωμένα δικαιώματα – ο χρήστης πρέπει να τα εκχωρήσει όλα ή κανένα.

Από τη στιγμή που εκχωρηθούν, τα δικαιώματα εφαρμόζονται στην εφαρμογή για όσο είναι εγκατεστημένη. Για να αποφευχθεί η σύγχυση από την πλευρά του χρήστη, το σύστημα δεν τον ειδοποιεί ξανά για τα δικαιώματα τα οποία έχουν εκχωρηθεί στην εφαρμογή, και οι εφαρμογές οι οποίες είναι προεγκατεστημένες δεν ζητούν εκχώρηση δικαιωμάτων από τον χρήστη. Τα δικαιώματα αφαιρούνται αν απεγκατασταθεί μία εφαρμογή, οπότε σε περίπτωση επαναγκατάστασης θα ξαναεμφανιστεί το σχετικό παράθυρο διαλόγου.

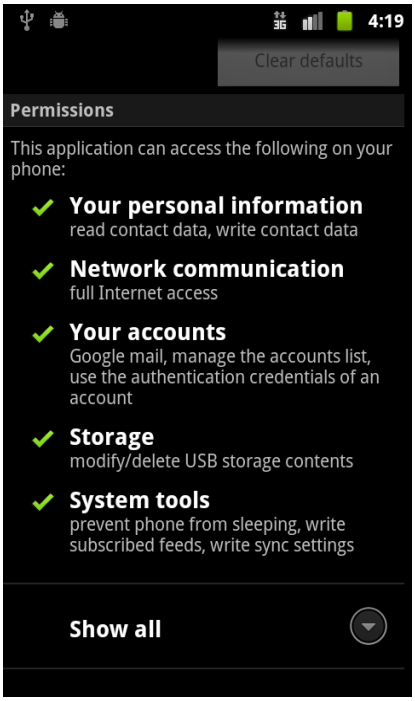
Μέσα από τις ρυθμίσεις της συσκευής, οι χρήστες μπορούν να δουν τα δικαιώματα που έχουν εκχωρηθεί σε εφαρμογές οι οποίες έχουν εγκατασταθεί. Οι χρήστες μπορούν επιπλέον να απενεργοποιήσουν καθολικά κάποιες λειτουργίες, όπως το GPS, το ραδιόφωνο, ή το wi-fi.

Σε περίπτωση που επιχειρήσει μία εφαρμογή να χρησιμοποιήσει κάποιο προστατευμένο χαρακτηριστικό, το οποίο δεν έχει δηλωθεί στο μανιφέστο της εφαρμογής, θα προκληθεί μία εξαίρεση ασφάλειας (security exception).

Θα πρέπει επίσης να σημειωθεί ότι οι εφαρμογές μπορούν να ορίσουν δικά τους δικαιώματα, τα οποία μπορούν να χρησιμοποιηθούν από τις ίδιες ή άλλες εφαρμογές.



Παράθυρο διαλόγου κατά την εγκατάσταση του Google Maps



Τα δικαιώματα που έχουν εκχωρηθεί στο gMail

Επικοινωνία μεταξύ των διεργασιών

Ορισμένες εφαρμογές επιχειρούν να υλοποιήσουν την επικοινωνία μεταξύ των διεργασιών χρησιμοποιώντας παραδοσιακές τεχνικές του Linux όπως είναι οι υποδοχές δικτύου και τα κοινόχρηστα αρχεία. Αντί αυτών των τεχνικών, οι σχεδιαστές του Android προτείνουν την χρήση των νέων μηχανισμών που περιλαμβάνει το Android όπως είναι τα Intents, Binders ή Messengers με ένα Service, και ο BroadcastReceiver. Οι μηχανισμοί που παρέχει το Android μας επιτρέπουν να επαληθεύσουμε την ταυτότητα της εφαρμογής η οποία συνδέεται στον μηχανισμό και να θέσουμε πολιτικές ασφάλειας για τον κάθε μηχανισμό.

Μεταδεδομένα Συσκευής

Το Android δίνει ιδιαίτερη σημασία στον περιορισμό της πρόσβασης σε δεδομένα τα οποία δεν είναι εν γένει ευαίσθητα, αλλά μπορεί εμμέσως να αποκαλύψουν χαρακτηριστικά για τον χρήστη, τις προτιμήσεις του, και τον τρόπο με τον οποίο χρησιμοποιεί την συσκευή.

Από προεπιλογή, οι εφαρμογές δεν έχουν πρόσβαση σε αρχεία καταγραφής, στο ιστορικό του προγράμματος περιήγησης, στον αριθμό του τηλεφώνου, ή σε πληροφορίες οι οποίες ταυτοποιούν το υλικό ή το δίκτυο. Αν μία εφαρμογή ζητήσει πρόσβαση σε αυτές τις πληροφορίες κατά την εγκατάσταση, το πρόγραμμα εγκατάστασης θα ζητήσει από τον χρήστη την απαραίτητη άδεια. Αν ο χρήστης δεν δώσει την συγκατάθεσή του, η εφαρμογή δεν θα εγκατασταθεί.

Υπογραφή Εφαρμογών (Application Signing)

Η υπογραφή εφαρμογών επιτρέπει στους προγραμματιστές να προσδιορίσουν τον δημιουργό μιας εφαρμογής και να αναβαθμίσουν την εφαρμογή τους χωρίς να απαιτείται η δημιουργία περίπλοκων διασυνδέσεων και δικαιωμάτων. Κάθε εφαρμογή η οποία τρέχει στο Android πρέπει να είναι υπογεγραμμένη από τον προγραμματιστή. Οι εφαρμογές που επιχειρούν να εγκατασταθούν χωρίς να είναι υπογεγραμμένες θα απορριφθούν είτε από το Google Play είτε από το πρόγραμμα εγκατάστασης της συσκευής.

Στο Google Play, η υπογραφή εφαρμογών γεφυρώνει την εμπιστοσύνη που έχει η Google με τον δημιουργό, και την εμπιστοσύνη που έχει ο δημιουργός με την εφαρμογή. Οι προγραμματιστές ξέρουν ότι η εφαρμογή τους παρέχεται χωρίς τροποποιήσεις στις συσκευές Android, και επιπλέον είναι εφικτό να λογοδοτήσουν για την συμπεριφορά της εφαρμογής τους.

Στο Android, η υπογραφή εφαρμογών είναι το πρώτο βήμα για την τοποθέτηση της εφαρμογής εντός του φίλτρου εφαρμογών. Το υπογεγραμμένο πιστοποιητικό της εφαρμογής καθορίζει ποιο αναγνωριστικό χρήστη είναι συσχετισμένο με ποια εφαρμογή. Διαφορετικές εφαρμογές τρέχουν με διαφορετικά αναγνωριστικά χρήστη. Η υπογραφή εφαρμογών διασφαλίζει ότι μία εφαρμογή δεν μπορεί να έχει πρόσβαση σε άλλη εφαρμογή παρά μόνο μέσω καλά ορισμένων μηχανισμών για την επικοινωνία μεταξύ των διεργασιών και των εφαρμογών.

Όταν μία εφαρμογή (ένα αρχείο apk) εγκαθίστανται σε μία συσκευή Android, ο διαχειριστής πακέτων ελέγχει κατά πόσον το apk έχει υπογραφεί κατάλληλα με το πιστοποιητικό το οποίο περιλαμβάνεται σε αυτό το apk. Αν το πιστοποιητικό (ή για την ακρίβεια, το δημόσιο κλειδί στο πιστοποιητικό) είναι ίδιο με το κλειδί το οποίο έχει χρησιμοποιηθεί για να υπογραφεί οποιοδήποτε άλλο apk στην συσκευή, το καινούριο apk έχει την επιλογή να ορίσει στο μανιφέστο ότι θα μοιράζεται το ίδιο αναγνωριστικό χρήστη με τα υπόλοιπα apk αρχεία που έχουν την ίδια υπογραφή.

Οι εφαρμογές μπορούν να υπογραφούν από κάποιον τρίτο (OEM για παράδειγμα) ή να είναι αυτό-υπογεγραμμένες. Το Android παρέχει την δυνατότητα για υπογραφή κώδικα με τη χρήση αυτό-υπογεγραμμένων πιστοποιητικών τα οποία οι προγραμματιστές μπορούν να δημιουργήσουν χωρίς να απαιτείται εξωτερική βοήθεια ή άδεια. Οι εφαρμογές δεν είναι απαραίτητο να είναι υπογεγραμμένες από κάποια κεντρική αρχή.

Οι εφαρμογές έχουν επίσης τη δυνατότητα να δηλώσουν δικαιώματα ασφάλειας με υπογραφή για το επίπεδο προστασίας (protectionLevel = “signature”), επιτρέποντας την πρόσβαση μόνο σε εφαρμογές οι οποίες έχουν υπογραφεί με το ίδιο κλειδί, με τις εφαρμογές να έχουν διαφορετικά αναγνωριστικά χρήστη (UIDs) και φίλτρα εφαρμογών. Μία πιο στενή σχέση, με χρήση κοινού φίλτρου εφαρμογών, είναι δυνατή μέσω του κοινού αναγνωριστικού χρήστη (shared UID), όπου δύο ή περισσότερες εφαρμογές υπογεγραμμένες με το ίδιο κλειδί μπορούν να δηλώσουν ένα κοινό αναγνωριστικό χρήστη στα μανιφέστα τους.

Δοκιμή (Testing)

Αυτοματοποιημένη δοκιμή

Στρατηγική δοκιμών στο Android

Η αυτοματοποιημένη δοκιμή των εφαρμογών είναι ιδιαίτερα σημαντική λόγω της μεγάλης ποικιλίας συσκευών που κυκλοφορούν στην αγορά. Επειδή δεν είναι δυνατόν να δοκιμάσουμε μία εφαρμογή με όλες τις δυνατές ρυθμίσεις μιας συσκευής, αποτελεί συνηθισμένη πρακτική η εκτέλεση δοκιμών με συνηθισμένες ρυθμίσεις.

Η εκτέλεση δοκιμών μας βοηθάει να βελτιώσουμε και να συντηρήσουμε μία εφαρμογή.

Πώς μπορούμε να δοκιμάσουμε τις εφαρμογές

Στο Android, οι δοκιμές βασίζονται στο JUnit. Μπορούμε να διαχωρίσουμε τις δοκιμές σε αυτές που χρειάζονται μόνο την JVM και σε αυτές που χρειάζονται το λειτουργικό σύστημα. Αν είναι εφικτό, είναι προτιμότερο να τρέξουμε τις δοκιμές απευθείας στην JVM, λόγω του ότι ο χρόνος που απαιτείται για την εκτέλεσή τους είναι πολύ μικρότερος σε σχέση με τον χρόνο που απαιτείται για την εγκατάσταση και την εκτέλεσή τους σε μία συσκευή Android.

Σημειώνεται ότι η εφαρμογή που δοκιμάζεται καλείται *εφαρμογή υπό δοκιμή*.

Δοκιμές μονάδων (unit tests) και δοκιμές ενσωμάτωσης (integration tests) στο Android

Η δοκιμή μονάδας δοκιμάζει μόνο τη λειτουργικότητα κάποιου στοιχείου.

Ας υποθέσουμε για παράδειγμα ότι ένα κουμπί σε μία Δραστηριότητα χρησιμοποιείται για την εκκίνηση μιας άλλης Δραστηριότητας. Μία δοκιμή μονάδας θα καθόριζε αν η αντίστοιχη πρόθεση (intent) δημιουργήθηκε σωστά, και όχι αν ξεκίνησε η δεύτερη Δραστηριότητα.

Μία δοκιμή ενσωμάτωσης θα έλεγχε επίσης, αν η Δραστηριότητα ξεκίνησε σωστά.

Το Android παρέχει διαφορετικές κλάσεις για δοκιμές μονάδων και δοκιμές ενσωμάτωσης.

Android και JUnit 3

Στη παρούσα φάση, το API δοκιμών του Android βασίζεται στο JUnit 3 και όχι στο JUnit 4.

Η Google εργάζεται για την αναβάθμιση του framework δοκιμών, με την οποία θα διαχωριστεί το framework δοκιμών από το framework εφαρμογών. Με αυτήν την αναβάθμιση σχεδιάζουν επίσης να αναβαθμίσουν το framework δοκιμών έτσι ώστε να βασίζεται στο JUnit 4.

Με το JUnit 3 οι κλάσεις δοκιμών κληρονομούν από την κλάση junit.framework.TestCase του JUnit 3.

Στο JUnit 3, οι μέθοδοι δοκιμών πρέπει να ξεκινούν με το πρόθεμα test. Πρέπει να καλείται η μέθοδος διαμόρφωσης setUp() και η τελική μέθοδος εκκαθάρισης teardown().

Τι πρέπει να υποβάλλουμε σε δοκιμή σε μία εφαρμογή

Ο ακόλουθος πίνακας περιλαμβάνει τις σημαντικές περιοχές τις οποίες πρέπει να υποβάλλουμε σε δοκιμή σε μία εφαρμογή.

Περιοχή υπό δοκιμή	Περιγραφή
Γεγονότα από τον κύκλο ζωής μιας Δραστηριότητας	Θα πρέπει να ελέγξουμε κατά πόσον οι Δραστηριότητες χειρίζονται καλά τα γεγονότα που συμβαίνουν κατά τον κύκλο ζωής τους. Θα πρέπει επίσης να ελέγξουμε αν η εφαρμογή συμπεριφέρεται σωστά όταν αλλάζουν οι ρυθμίσεις (configuration), και αν επανέρχονται στην πρότερη κατάσταση (state) τα στοιχεία της διεπαφής χρήστη.
Λειτουργίες του συστήματος αρχείων και της βάσης δεδομένων	Η ανάγνωση και η εγγραφή από και προς το σύστημα αρχείων θα πρέπει να ελεγχθεί, συμπεριλαμβανομένων των λειτουργιών των βάσεων δεδομένων.
Διαφορετικές ρυθμίσεις της συσκευής	Θα πρέπει επίσης να ελέγξουμε κατά πόσον η εφαρμογή συμπεριφέρεται καλά με διαφορετικές ρυθμίσεις της συσκευής.

Προϋποθέσεις δοκιμής

Στο Android αποτελεί καλή πρακτική να έχουμε μία μέθοδο με όνομα testPreconditions() η οποία δοκιμάζει (tests) τις προϋποθέσεις όλων των δοκιμών. Αν αυτή η μέθοδος αποτύχει, αυτό σημαίνει ότι δεν ισχύουν οι υποθέσεις στις οποίες βασίζονται οι δοκιμές.

Για ποιες δοκιμές απαιτείται το λειτουργικό σύστημα προκειμένου να τρέξουν

Δοκιμή τυπικών κλάσεων της Java

Αν οι κλάσεις δεν καλούν το Android API, μπορούμε να χρησιμοποιήσουμε το framework δοκιμών JUnit (ή οποιοδήποτε άλλο framework δοκιμών για Java) χωρίς περιορισμούς.

Το πλεονέκτημα αυτής της μεθόδου είναι ότι μπορούμε να χρησιμοποιήσουμε οποιοδήποτε framework δοκιμών για Java και ότι η ταχύτητα εκτέλεσης της δοκιμής μονάδας είναι πολύ μεγάλη σε σχέση με την ταχύτητα που έχουν οι δοκιμές που χρειάζονται το λειτουργικό σύστημα.

Δοκιμή Java κλάσεων οι οποίες χρησιμοποιούν το API του Android

Αν θέλουμε να δοκιμάσουμε κώδικα ο οποίος χρησιμοποιεί το API του Android, θα πρέπει να τρέξουμε αυτές τις δοκιμές σε μία συσκευή Android. Δυστυχώς, αυτό κάνει την εκτέλεση των δοκιμών να κρατάει περισσότερο.

Πρότζεκτ δοκιμών στο Android και διενέργεια δοκιμών

Πρότζεκτ δοκιμών

Ο προτιμώμενος τρόπος οργάνωσης δοκιμών είναι να διατηρούνται σε ξεχωριστά πρότζεκτ δοκιμών ή σε ξεχωριστούς φακέλους προέλευσης. Το ADT plug-in για το Eclipse δίνει έμφαση στη χρήση διαφορετικών πρότζεκτ.

Αντί για στοιχεία Android (Android components), μία εφαρμογή δοκιμών για Android περιέχει μία ή περισσότερες κλάσεις δοκιμών.

Το πρότζεκτ υπό δοκιμή πρέπει να εισαχθεί σαν εξάρτηση (dependency) στο πρότζεκτ δοκιμών. Το αρχείο AndroidManifest.xml πρέπει επίσης να αναφέρει ότι το πρότζεκτ δοκιμών χρησιμοποιεί την βιβλιοθήκη android.test.runner και επιπλέον να αναφέρει τον εκτελεστή δοκιμών (test runner) για την δοκιμή μονάδας.

Ένα πρότζεκτ δοκιμών προσδιορίζει επίσης το πακέτο της εφαρμογής που θα δοκιμαστεί με την ιδιότητα *android:targetPackage.* Ακολουθεί ένα παράδειγμα AndroidManifest.xml για ένα πρότζεκτ δοκιμών.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="de.vogella.android.test.target.test"
    android:versionCode="1"
    android:versionName="1.0">
  <uses-sdk android:minSdkVersion="8" />
  <instrumentation
     android:targetPackage="de.vogella.android.test.target"
     android:name="android.test.InstrumentationTestRunner" />
  <application android:icon="@drawable/icon" android:label="@string/app_name">
```

```
    <uses-library android:name="android.test.runner" />
    </application>
</manifest>
```

Δημιουργία ενός πρότζεκτ δοκιμών με το ADT

Το ADT παρέχει υποστήριξη για την δημιουργία πρότζεκτ δοκιμών μέσω ενός οδηγού (wizard) δημιουργίας πρότζεκτ. Μπορούμε να ανοίξουμε αυτόν τον οδηγό από το File → New → Other... → Android → Android Test Project.

Εκτέλεση δοκιμής

Για να εκκινήσουμε μία δοκιμή, θα πρέπει να κάνουμε δεξί κλικ στην κλάση δοκιμών και να επιλέξουμε Run As → Android JUnit Test.

Αντικείμενα mock

Το Android παρέχει κλάσεις για δημιουργία αντικειμένων mock στα πακέτα android.test και android.test.mock.

Ακολουθεί μία λίστα με τα σημαντικότερα αντικείμενα mock τα οποία μπορούν να χρησιμοποιηθούν σε δοκιμές.

- MockApplication
- MockContext
- MockResources
- MockContentProvider
- MockContentResolver
- MockPackageManager

Δοκιμή του αντικειμένου Application

Η κλάση application περιέχει την λογική, τα δεδομένα και τις ρυθμίσεις που σχετίζονται με ολόκληρη την εφαρμογή. Συνεπώς, θα πρέπει να δοκιμάσουμε αυτό το αντικείμενο για να διασφαλίσουμε ότι δουλεύει σωστά.

Για αυτόν τον σκοπό μπορούμε να χρησιμοποιήσουμε την κλάση ApplicationTestCase.

Δοκιμή Υπηρεσιών (Service testing)

Για να δοκιμάσουμε μία υπηρεσία μπορούμε να χρησιμοποιήσουμε την κλάση ServiceTestCase.

Δοκιμή Παρόχων Περιεχομένου

Για να δοκιμάσουμε έναν πάροχο περιεχομένου μπορούμε να χρησιμοποιήσουμε την κλάση ProviderTestCase2.

Άγκιστρα

Instrumentation

Το API για τις δοκιμές παρέχει άγκιστρα στον κύκλο ζωής των στοιχείων λογισμικού και της εφαρμογής. Αυτά τα άγκιστρα απαρτίζουν το instrumentation API και επιτρέπουν στις δοκιμές να ελέγχουν γεγονότα από τον κύκλο ζωής (life cycle events) και την αλληλεπίδραση του χρήστη.

Υπό κανονικές συνθήκες μία εφαρμογή μπορεί μόνο να αντιδράσει σε τέτοια γεγονότα. Για παράδειγμα, όταν το Android δημιουργεί μία Δραστηριότητα καλείται η μέθοδος onCreate() για την Δραστηριότητα. Άλλη περίπτωση είναι όταν ο χρήστης πατήσει κάποιο κουμπί ή πλήκτρο οπότε και καλείται ο αντίστοιχος κώδικας. Μέσω του instrumentation μπορούμε να προκαλέσουμε τέτοια γεγονότα.

Μόνο μία κλάση δοκιμών που χρησιμοποιεί το instrumentation API μας επιτρέπει να στείλουμε γεγονότα πλήκτρων (key events) ή αφής (touch events) στην εφαρμογή υπό δοκιμή.

Για παράδειγμα, μία δοκιμή μπορεί να καλέσει την μέθοδο getActivity() η οποία εκκινεί μία Δραστηριότητα και επιστρέφει την Δραστηριότητα υπό δοκιμή. Έπειτα, μπορούμε να καλέσουμε την μέθοδο finish() και εν συνεχεία την μέθοδο getActivity() και πάλι, για να ελέγξουμε εάν η εφαρμογή επανάφερε σωστά την κατάσταση της Δραστηριότητας.

Το instrumentation API μας επιτρέπει να τρέξουμε το πρότζεκτ δοκιμών και το κανονικό πρότζεκτ στην ίδια διεργασία προκειμένου το πρότζεκτ δοκιμών να μπορεί να καλεί απευθείας μεθόδους του πρότζεκτ υπό δοκιμή.

Πώς διενεργεί δοκιμές το Android

Οι κλάσεις του συστήματος φορτώνουν και εκκινούν το πακέτο δοκιμών (test package), τερματίζουν τυχόν διεργασίες οι οποίες τρέχουν κάποιο στιγμιότυπο της εφαρμογής υπό δοκιμή, και στη συνέχεια φορτώνουν ένα νέο στιγμιότυπο της εφαρμογής υπό δοκιμή. Ακολούθως, δίνουν τον έλεγχο στον InstrumentationTestRunner, ο οποίος τρέχει κάθε κλάση δοκιμών που υπάρχει στο πακέτο δοκιμών.

Ούτε οι κλάσεις του συστήματος, ούτε ο InstrumentationTestRunner τρέχουν την εφαρμογή υπό δοκιμή. Αυτό το κάνουν οι κλάσεις δοκιμών απευθείας. Είτε καλούν μεθόδους στην εφαρμογή υπό δοκιμή, είτε καλούν τις μεθόδους που παρέχονται από το instrumentation API, οι οποίες προκαλούν γεγονότα από τον κύκλο ζωής και την αλληλεπίδραση του χρήστη, στην εφαρμογή υπό δοκιμή.

Δοκιμή Δραστηριοτήτων

Κύκλος ζωής δραστηριοτήτων και instrumentation

Όταν χρησιμοποιούμε το instrumentation API για να δοκιμάσουμε δραστηριότητες, οι μέθοδοι από τον κύκλο ζωής δεν καλούνται αυτόματα, μόνο η μέθοδος onCreate() καλείται αν καλέσουμε την μέθοδο startActivity(). Μπορούμε να καλέσουμε απευθείας τις υπόλοιπες μεθόδους μέσω των βοηθητικών μεθόδων getInstrumentation().callActivityOn*.

Δοκιμές μονάδων για Δραστηριότητες

Για να δοκιμάσουμε μία Δραστηριότητα σε απομόνωση, μπορούμε να χρησιμοποιήσουμε την κλάση ActivityUnitTestCase.

Αυτή η κλάση μας επιτρέπει να ελέγξουμε την διάταξη καθώς και απομονωμένες μεθόδους μιας Δραστηριότητας, όπως επίσης να ελέγξουμε αν οι προθέσεις (intents) δημιουργούνται κανονικά. Η πρόθεση δεν στέλνεται στο σύστημα, αλλά μπορούμε να χρησιμοποιήσουμε την μέθοδο getStartedActivityIntent() για να αποκτήσουμε πρόσβαση σε μία πιθανή πρόθεση και να επαληθεύσουμε τα δεδομένα.

Η κλάση ActivityUnitTestCase εκκινεί την Δραστηριότητα σε ένα IsolatedContext, δηλαδή η Δραστηριότητα είναι απομονωμένη ως επί το πλείστον από το σύστημα.

Εφόσον αυτή η δοκιμή τρέχει σε ένα IsolatedContext, η δοκιμή πρέπει να εκκινήσει την Δραστηριότητα. Η Δραστηριότητα δεν εκκινείται αυτόματα από το σύστημα.

```
Intent intent = new Intent(getInstrumentation().getTargetContext(), MainActivity.class);
startActivity(intent, null, null);

// μετά από αυτήν την κλήση μπορούμε να πάρουμε την Δραστηριότητα με την μέθοδο
// getActivity()
```

Δοκιμές ενσωμάτωσης για Δραστηριότητες

Λειτουργικές δοκιμές για μία Δραστηριότητα μπορούν να γραφούν με την κλάση ActivityInstrumentationTestCase2. Αυτή η δοκιμή χρησιμοποιεί την πλήρη υποδομή του συστήματος Android και μας επιτρέπει να αλληλεπιδράσουμε με διαφορετικά στοιχεία. Η επικοινωνία με την υποδομή του συστήματος γίνεται μέσω της κλάσης Instrumentation στην οποία μπορούμε να αποκτήσουμε πρόσβαση μέσω της μεθόδου getInstrumentation(). Αυτή η κλάση μας επιτρέπει να στείλουμε γεγονότα πληκτρολογίου και γεγονότα κλικ.

Αν επιθυμούμε να θέσουμε τιμές απευθείας, πρέπει να χρησιμοποιήσουμε την μέθοδο runOnUiThread() της Δραστηριότητας, δεδομένου ότι μόνο το νήμα διεπαφής χρήστη (UI thread) μπορεί να τροποποιήσει την διεπαφή χρήστη. Αν όλες οι προτάσεις εντός της μεθόδου αλληλεπιδρούν με το νήμα διεπαφής

χρήστη, μπορούμε να χρησιμοποιήσουμε το σχόλιο @UiThreadTest στην μέθοδο. Σε αυτήν την περίπτωση δεν μπορούμε να χρησιμοποιήσουμε μεθόδους οι οποίες δεν τρέχουν στο κύριο νήμα / νήμα διεπαφής χρήστη.

Οι δοκιμές οι οποίες βασίζονται στην ActivityInstrumentationTestCase2 εκκινούν την Δραστηριότητα στο τυπικό context, όπως όταν ένας χρήστης εκκινεί την εφαρμογή.

Μπορούμε να χρησιμοποιήσουμε την Instrumentation.invokeMenuActionSync(context, MENU_ID,0) για να προκαλέσουμε μία ενέργεια στο μενού.

Δοκιμή της αρχικής κατάστασης

Αποτελεί καλή πρακτική η δοκιμή της αρχικής κατάστασης της εφαρμογής πριν ξεκινήσει η βασική Δραστηριότητα, έτσι ώστε να είμαστε σίγουροι ότι πληρούνται οι συνθήκες δοκιμής της Δραστηριότητας.

Δοκιμές διαχείρισης της κατάστασης

Προτείνεται να γράψουμε δοκιμές οι οποίες ελέγχουν αν διατηρείται η κατάσταση μιας Δραστηριότητας ακόμα και αν αυτή έχει παυθεί ή τερματιστεί από το σύστημα.

Η κλάση ActivityInstrumentationTestCase2 χρησιμοποιεί την κλάση Instrumentation, η οποία μας επιτρέπει να καλούμε απευθείας τα άγκιστρα στον κύκλο ζωής των Δραστηριοτήτων. Για παράδειγμα, μπορούμε να καλέσουμε τις μεθόδους onPause() και onDestroy(), και στη συνέχεια την onCreate() για να ελέγξουμε αν η κατάσταση διατηρείται.

Δοκιμές της διεπαφής χρήστη

Συνολική δοκιμή της διεπαφής χρήστη

Οι λειτουργικές δοκιμές της διεπαφής χρήστη δοκιμάζουν ολόκληρη την εφαρμογή και όχι μεμονωμένα στοιχεία της.

uiautomator

Το Android SDK περιέχει την βιβλιοθήκη Java *uiautomator* για την δημιουργία δοκιμών της διεπαφής χρήστη και επιπλέον παρέχει μία μηχανή για την εκτέλεση αυτών των δοκιμών. Τα εργαλεία αυτά είναι διαθέσιμα από το API 16 και μετά.

Τα πρότζεκτ δοκιμών που χρησιμοποιούν το uiautomator αποτελούν μεμονωμένα πρότζεκτ Java. Χρησιμοποιούν την βιβλιοθήκη JUnit 3 και επιπλέον τα αρχεία uiautomator.jar και android.jar από τον φάκελο android-sdk/platforms/api-version πρέπει να προστεθούν στη διαδρομή δόμησης (build path).

Το uiautomator παρέχει την κλάση UiDevice για την επικοινωνία με την συσκευή, την κλάση UiSelector για αναζήτηση στοιχείων στην οθόνη, και το UiObject για τα διάφορα στοιχεία της διεπαφής χρήστη. Η κλάση UiCollection επιτρέπει την ταυτόχρονη επιλογή ενός αριθμού στοιχείων της διεπαφής χρήστη, και η κλάση UiScrollable επιτρέπει την κύλιση (scroll) σε κάποιο View για την εύρεση κάποιου στοιχείου.

Google Services

Αξιοποιώντας τα Google Services μπορούμε να εμπλουτίσουμε εύκολα την εφαρμογή μας με διάφορα χαρακτηριστικά, προκειμένου να προσελκύσουμε χρήστες ενός μεγαλύτερου εύρους συσκευών, χρησιμοποιώντας γνωστές υπηρεσίες της Google. Ακολουθεί μία αναφορά ορισμένων εξ' αυτών:

Games: Με τα Google Play Game Services μπορούμε να δώσουμε στους χρήστες των παιχνιδιών μας μια πιο κοινωνική εμπειρία. Υπάρχει η δυνατότητα προσθήκης επιτευγμάτων, πινάκων με τους κορυφαίους παίκτες (leaderboards), multiplayer σε πραγματικό χρόνο, και άλλων δημοφιλών χαρακτηριστικών. Οι παίκτες μπορούν να εισέρχονται χρησιμοποιώντας τα διαπιστευτήρια χρήστη που έχουν στο Google+ και να μοιράζονται την εμπειρία τους από το παιχνίδι με φίλους.

Location: Με τα location APIs είναι ευκολότερη η δημιουργία εφαρμογών που γνωρίζουν τη θέση του χρήστη, εφόσον δεν χρειάζεται ο προγραμματιστής να επικεντρωθεί στις λεπτομέρειες της τεχνολογίας που χρησιμοποιείται για τον εντοπισμό της θέσης (location technology). Επιπλέον, επιτρέπουν την ελαχιστοποίηση της κατανάλωσης ενέργειας χρησιμοποιώντας όλες τις δυνατότητες του υλικού της συσκευής.

- *Geofencing:*
  Επιτρέπει στην εφαρμογή μας να θέσει γεωγραφικά όρια γύρω από συγκεκριμένες τοποθεσίες και να δέχεται ειδοποιήσεις όταν ο χρήστης μπαίνει ή φεύγει από αυτές τις περιοχές.
- *Activity recognition:*
  Είναι χρήσιμο για μία εφαρμογή να γνωρίζει τι κάνει ο χρήστης εκείνη την στιγμή προκειμένου να εμφανίσει το κατάλληλο περιεχόμενο. Με το Activity recognition API είναι εύκολος ο έλεγχος της τρέχουσας δραστηριότητας του χρήστη—κάθεται ακίνητος, περπατάει, κάνει ποδήλατο, είναι σε κάποιο όχημα—με πολύ αποτελεσματική χρήση της μπαταρίας.

Maps: Το Google Maps API μας επιτρέπει να χρησιμοποιήσουμε στην εφαρμογή μας τους χάρτες που παρέχει η Google και να τους προσαρμόσουμε με δείκτες (markers), να ελέγξουμε την οπτική του χρήστη, και να σχεδιάσουμε γραμμές και σχήματα.

Google+: Χρησιμοποιώντας το Google+ για το Android, μπορούμε να πιστοποιήσουμε τον χρήστη της εφαρμογής μας. Αφού γίνει η πιστοποίηση, μπορούμε επιπλέον να έχουμε πρόσβαση στο δημόσιο προφίλ και στο κοινωνικό γράφημα του χρήστη.

Drive: Με το Google Drive Android API μπορούμε να δώσουμε στους χρήστες πρόσβαση στα αρχεία τους, όπου και αν βρίσκονται, από οποιαδήποτε συσκευή. Το API κάνει εύκολη την αποθήκευση και τον συγχρονισμό των αρχείων του χρήστη μεταξύ της συσκευής και του cloud.
Google Play In-app Billing: Με το Google Play In-app Billing είναι εφικτή η πώληση ψηφιακού περιεχομένου από τις εφαρμογές μας. Αυτό το ψηφιακό περιεχόμενο μπορεί να πωληθεί με μία και μοναδική χρέωση ή με διαφορετικές χρεώσεις οι οποίες μπορεί να γίνονται και μέσω συνδρομών. Μπορούμε να χρησιμοποιήσουμε αυτήν την υπηρεσία για να πουλήσουμε μία μεγάλη ποικιλία περιεχομένου, στο οποίο περιλαμβάνεται περιεχόμενο το οποίο μπορεί να κατεβάσει ο χρήστης, όπως αρχεία πολυμέσων ή φωτογραφίες, εικονικό περιεχόμενο όπως επίπεδα παιχνιδιών ή φίλτρα, premium υπηρεσίες και χαρακτηριστικά κ.ά.
Ads: Το Google Mobile Ads επιτρέπει την ενσωμάτωση διαφημίσεων σε μία εφαρμογή. Υποστηρίζει μία μεγάλη ποικιλία διαφημίσεων χάρη στο μεγάλο όγκο των διαφημιστών που συνεργάζονται με την Google (πάνω από ένα εκατομμύριο). Υπάρχει η δυνατότητα προβολής διαφημίσεων με βάση την τοποθεσία που βρίσκεται ο χρήστης (location-based ads), οι οποίες μπορούν να αποφέρουν περισσότερα έσοδα σε σχέση με τις «συμβατικές» διαφημίσεις.
Google Cloud Messaging (**GCM**): Το GCM για Android επιτρέπει την επικοινωνία ανάμεσα στον διακομιστή μας και την εφαρμογή μέσω ασύγχρονων μηνυμάτων. Δεν χρειάζεται να μας απασχολεί ο χειρισμός χαμηλού επιπέδου (low level) πτυχών αυτής της επικοινωνίας, όπως είναι η κατασκευή των μηνυμάτων και της ουράς. Χρησιμοποιώντας αυτήν την υπηρεσία, μπορούμε να υλοποιήσουμε εύκολα ένα σύστημα ειδοποιήσεων για την εφαρμογή μας.

Το 2012 η Google άρχισε να διαχωρίζει τα Google Services από το Android, παρέχοντας με αυτό τον τρόπο τη δυνατότητα αναβάθμισης τους από το Google Play Store χωρίς να απαιτείται αναβάθμιση του λειτουργικού.

Πώς δουλεύουν τα Google Play Services

Όταν η Google παρουσίασε τα Google Play Services στο Google I/O 2012, είπε ότι η πλατφόρμα αποτελείται από ένα στοιχείο υπηρεσιών το οποίο τρέχει στην συσκευή και μία βιβλιοθήκη πελάτη (client library) η οποία περιλαμβάνεται στην εφαρμογή.
Αυτό σημαίνει ότι τα Google Play Services δουλεύουν χάρη σε δύο βασικά στοιχεία: την βιβλιοθήκη πελάτη και το Google Play Services APK.

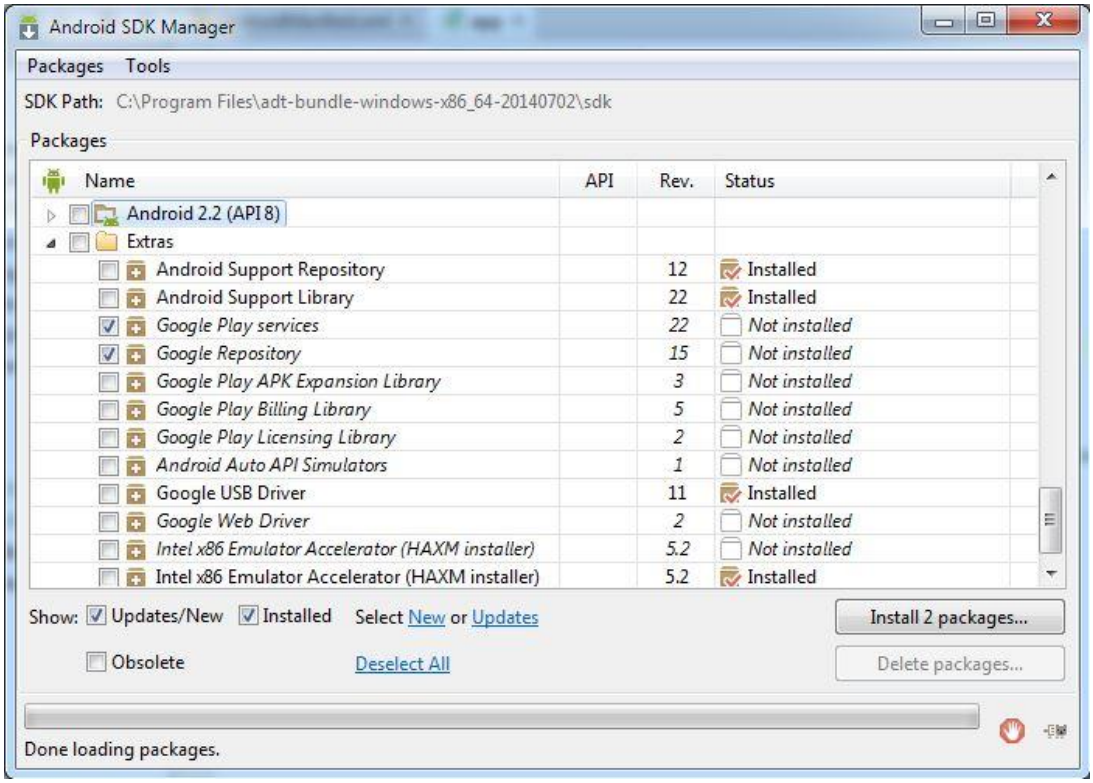
Βιβλιοθήκη πελάτη: Η βιβλιοθήκη πελάτη των Google Play Services περιλαμβάνει τις διασυνδέσεις (interfaces) που χρησιμοποιούνται από την εφαρμογή για κάθε Google Service. Η βιβλιοθήκη επιτρέπει στους χρήστες να εξουσιοδοτήσουν την εφαρμογή με πρόσβαση σε αυτά τα services χρησιμοποιώντας τα διαπιστευτήριά τους. Η βιβλιοθήκη πελάτη αναβαθμίζεται από την Google με καινούρια χαρακτηριστικά και υπηρεσίες. Μπορούμε να αναβαθμίσουμε την βιβλιοθήκη παρέχοντας μία νεότερη έκδοση της εφαρμογής

μας, το οποίο βέβαια δεν είναι απαραίτητο εάν δεν συμπεριλαμβάνουμε τα νεότερα χαρακτηριστικά που υποστηρίζει.
Google Play Services APK: Το Google Play Services Android Package (APK) τρέχει παρασκηνιακά (background) σαν υπηρεσία στο Android. Χρησιμοποιώντας την βιβλιοθήκη πελάτη, η εφαρμογή αποκτάει πρόσβαση σε αυτήν την υπηρεσία, η οποία είναι αυτή που εκτελεί τις διάφορες ενέργειες ενώ τρέχει η εφαρμογή. Το APK δεν είναι εγγυημένο ότι θα είναι εγκατεστημένο σε όλες τις συσκευές. Σε περίπτωση που δεν είναι εγκατεστημένο μπορεί κάποιος χρήστης να το προμηθευτεί από το Google Play Store. Κατ' αυτό τον τρόπο, δεν χρειάζεται να αναβαθμίζουμε την εφαρμογή μας κάθε φορά που αναβαθμίζονται τα Google Play Services από την Google. Εφόσον το Google Play services APK παρέχεται μέσω του Google Play Store, οι αναβαθμίσεις που δέχονται οι υπηρεσίες δεν εξαρτώνται από τις αναβαθμίσεις που γίνονται στο λειτουργικό από τους κατασκευαστές ή τους παρόχους κινητής τηλεφωνίας. Προκειμένου να αναβαθμίζεται αυτόματα το Google Play Services APK μέσω του Google Play Store, πρέπει να είναι εγκατεστημένη η εφαρμογή Google Play Store και η έκδοση του Android να είναι η 2.3 ή κάποια νεότερη. Μολονότι τα Google Play Services δεν αποτελούν μέρος της πλατφόρμας Android, υποστηρίζονται από τις περισσότερες συσκευές Android. Όποια συσκευή έχει εγκατεστημένη την έκδοση 2.2 του Android ή κάποια νεότερη, μπορεί να εγκαταστήσει οποιαδήποτε εφαρμογή που κάνει χρήση των Google Play Services.

Προσθήκη των Google Play Services στο Android Studio

Για να χρησιμοποιήσουμε τα Google Play Services στην εφαρμογή μας θα πρέπει να προσθέσουμε την βιβλιοθήκη πελάτη. Η βιβλιοθήκη είναι διαθέσιμη μέσω του Android SDK Manager. Αφού ανοίξουμε τον SDK Manager από το Tools → Android, ανοίγουμε τον φάκελο Extras και επιλέγουμε τα:

- Google Repository
- Google Play Services



Με αυτόν τον τρόπο η βιβλιοθήκη θα προστεθεί στον φάκελο όπου έχουμε εγκαταστήσει το Android SDK /sdk/extras/google/google_play_services/.

Στη συνέχεια πρέπει να κάνουμε τα ακόλουθα:

1. Ανοίγουμε το αρχείο build.gradle.
2. Προσθέτουμε έναν νέο κανόνα δόμησης στα dependencies για την τελευταία έκδοση των play-services. Για παράδειγμα:

```
apply plugin: 'com.android.application'
...

dependencies {
    compile 'com.android.support:appcompat-v7:21.0.3'
    compile 'com.google.android.gms:play-services:6.5.87'
}
```

3. Σώζουμε τις αλλαγές και κάνουμε κλικ στο **Sync Project with Gradle Files** στην μπάρα εργαλείων.
4. Ανοίγουμε το μανιφέστο και προσθέτουμε την ακόλουθη ετικέτα σαν θυγατρικό στοιχείο του στοιχείου <application>:

```
<meta-data android:name="com.google.android.gms.version"
           android:value="@integer/google_play_services_version" />
```

Movie Showtimes

Η εφαρμογή Movie Showtimes η οποία αναπτύχθηκε σε γλώσσα Java δίνει στον χρήστη τη δυνατότητα να δει το πρόγραμμα των ταινιών σε διάφορες πόλεις της χώρας, όπως επίσης και να πάρει κάποιες πληροφορίες για τις ταινίες που προβάλλονται, όπως είναι η πλοκή, το είδος της ταινίας, η βαθμολογία που έχει συγκεντρώσει η ταινία από τις αξιολογήσεις των χρηστών κ.ά.

Οι πληροφορίες αντλούνται από τον ιστότοπο www.cine.gr. Σε πρώτη φάση, έπρεπε να σχεδιαστεί ένα πρόγραμμα (HtmlParsing) το οποίο θα αποθηκεύει τις πληροφορίες αυτές σε μία βάση δεδομένων, έτσι ώστε να έχει πρόσβαση σε αυτές η εφαρμογή. Για λόγους ασφάλειας, η εφαρμογή δεν επικοινωνεί απευθείας με τη βάση δεδομένων. Σε αυτήν την περίπτωση τα διαπιστευτήρια διαχείρισης (administrator credentials) θα ήταν διαθέσιμα σε κάποιον κακόβουλο χρήστη μέσω αποσυμπίλησης (reverse engineer). Η επικοινωνία με τη βάση δεδομένων γίνεται μέσω ενός web api το οποίο υλοποιείται σε php.

HtmlParsing

Ο αναλυτής Html (Html parser) είναι γραμμένος σε γλώσσα Java. Χρησιμοποιήθηκε το jsoup (jsoup.org), το οποίο είναι μία βιβλιοθήκη γραμμένη σε Java, η οποία διευκολύνει σημαντικά την επεξεργασία του Html κώδικα. Η δομή της βάσης στην οποία εισάγονται τελικά τα δεδομένα είναι η ακόλουθη:

Locations (Name, LastUpdated)

Theaters (Name, AddressAndPhoneNumber, LocationName)

Showings (MovieOriginalTitle, MovieReleaseYear, TheaterName, AddressAndPhoneNumber, Hours)

Movies (OriginalTitle, ReleaseYear, TranslatedTitle, Genre, Duration, SuitableUrl, SuitableMessage, Premiere, Rating, PosterUrl, Plot)

Σε μία πόλη, μπορεί να υπάρχουν περισσότερες από μία αίθουσες. Άρα χρειάζεται μία συσχέτιση «1 προς πολλά» από τον πίνακα Locations στον πίνακα Theaters, οπότε το κλειδί του πίνακα Locations μπαίνει σαν ξένο κλειδί στον πίνακα Theaters. Επιπλέον, σε μία αίθουσα μπορεί να προβάλλονται περισσότερες από μία ταινίες, ενώ μία ταινία μπορεί να προβάλλεται σε πολλές αίθουσες. Άρα χρειάζεται μία συσχέτιση «πολλά προς πολλά» από τον πίνακα Theaters στον πίνακα Movies και ένας επιπλέον πίνακας για να ικανοποιηθεί αυτή η συσχέτιση, ο πίνακας Showings, ο οποίος περιλαμβάνει και το πεδίο Hours (ημέρες και ώρες προβολών).

Στο www.cine.gr υπάρχει μία λίστα με πόλεις. Επιλέγοντας κάποια από αυτές ο χρήστης μεταβαίνει στο πρόγραμμα προβολών για αυτήν την πόλη, ενώ κάνοντας κλικ σε κάποια ταινία πηγαίνει στην σελίδα όπου υπάρχουν πληροφορίες για αυτήν.

Το πρόγραμμα εισάγει τα δεδομένα σε πίνακες String πριν τελικά εισαχθούν στη βάση. Αποτελείται από τις ακόλουθες κλάσεις και συναρτήσεις:

LocationsExtractor. Η κλάση αυτή έχει μία μόνο συνάρτηση, την getLocations, η οποία επιστρέφει έναν πίνακα String δύο διαστάσεων. Για την κάθε πόλη ο πίνακας περιλαμβάνει τις ακόλουθες πληροφορίες: το όνομα της πόλης, την ημερομηνία που έγινε η ανανέωση του προγράμματος προβολών η οποία είναι διαφορετική για κάθε πόλη, και το url στο οποίο βρίσκεται το πρόγραμμα των προβολών. Το url αυτό δεν αποθηκεύεται στην βάση. Χρειάζεται όμως για να εξαχθούν τα δεδομένα από την εν λόγω σελίδα. Από αυτόν τον πίνακα String εισάγονται τα δεδομένα στον πίνακα Locations.

ShowingsExtractor. Η κλάση αυτή περιλαμβάνει 4 συναρτήσεις.

Η συνάρτηση getMovieLinks παίρνει σαν παράμετρο τον πίνακα String που επιστρέφει η getLocations, και επιστρέφει έναν πίνακα String δύο διαστάσεων ο οποίος για την κάθε προβαλλόμενη ταινία περιλαμβάνει το url στο οποίο βρίσκονται οι πληροφορίες για αυτήν, όπως επίσης και τον πρωτότυπο τίτλο της ταινίας. Το url δεν αποθηκεύεται στην βάση. Χρειάζεται για να εξαχθούν τα δεδομένα από αυτήν την σελίδα.

Η συνάρτηση getShowingsAlternate παίρνει σαν παράμετρο τον πίνακα String που επιστρέφει η getLocations, και επιστρέφει έναν πίνακα String δύο διαστάσεων ο οποίος στο κάθε κελί αποθηκεύει τα εξής: την τοποθεσία, το url στο οποίο βρίσκονται οι πληροφορίες για την ταινία, τον πρωτότυπο τίτλο της ταινίας, το όνομα της αίθουσας, την διεύθυνση και το τηλέφωνο της αίθουσας, τις μέρες και ώρες στις οποίες προβάλλεται η ταινία.

Η συνάρτηση getShowings παίρνει σαν παράμετρο τον πίνακα String που επιστρέφει η getLocations, καθώς και τον πίνακα String που επιστρέφει η συνάρτηση getMovies της κλάσης MoviesExtractor. Επιστρέφει έναν πίνακα String δύο διαστάσεων ο οποίος περιλαμβάνει για την κάθε προβολή τα εξής: τον πρωτότυπο τίτλο της ταινίας, το έτος κυκλοφορίας, το όνομα της αίθουσας, την διεύθυνση και το τηλέφωνο της αίθουσας, τις μέρες και ώρες στις οποίες προβάλλεται η ταινία. Από αυτόν τον πίνακα String εισάγονται τα δεδομένα στον πίνακα Showings.

Η συνάρτηση getTheaters παίρνει σαν παράμετρο τον πίνακα String που επιστρέφει η getLocations, και επιστρέφει έναν πίνακα String δύο διαστάσεων, ο οποίος αποθηκεύει τα εξής στο κάθε κελί: το όνομα της αίθουσας, την διεύθυνση και το τηλέφωνο της αίθουσας, και την τοποθεσία στην οποία βρίσκεται. Από αυτόν τον πίνακα String εισάγονται τα δεδομένα στον πίνακα Theaters.

MoviesExtractor. Η κλάση αυτή περιλαμβάνει μόνο μία συνάρτηση, την getMovies. Η getMovies παίρνει σαν παράμετρο τον πίνακα που επιστρέφει η getMovieLinks και επιστρέφει έναν πίνακα String δύο διαστάσεων ο οποίος αποθηκεύει διάφορες πληροφορίες για την κάθε ταινία, όπως αυτές φαίνονται στον πίνακα Movies. Από αυτόν τον πίνακα String εισάγονται τα δεδομένα στον πίνακα Movies.

HtmlParsing. Η κλάση αυτή περιλαμβάνει την συνάρτηση main.

Η σύνδεση με τη βάση γίνεται με JDBC. Τα δεδομένα των πινάκων διαγράφονται και στην συνέχεια γίνεται η εισαγωγή. Το πρόγραμμα των προβολών αλλάζει κάθε μία εβδομάδα.

```
import java.sql.*;

public class HtmlParsing {
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL =
"jdbc:mysql://sql3.freesqldatabase.com/sql346520?useUnicode=yes&chara
cterEncoding=UTF-8";
   static final String USER = "sql346520";
   static final String PASS = "sF8*yJ4!";
   static Connection connection = null;
   static Statement statement = null;

   public static void main(String args[]) throws Exception {
      LocationsExtractor locExtractor = new LocationsExtractor();
      ShowingsExtractor shExtractor = new ShowingsExtractor();
      MoviesExtractor mExtractor = new MoviesExtractor();

      String locations[][] = locExtractor.getLocations();
      String movieLinks[][] = shExtractor.getMovieLinks(locations);
      String movies[][] = mExtractor.getMovies(movieLinks);
      String showings[][] = shExtractor.getShowings(locations,
movies);
      String theaters[][] = shExtractor.getTheaters(locations);

      Class.forName(JDBC_DRIVER);
      System.out.println("Connecting to the database...");
      connection = DriverManager.getConnection(DB_URL, USER, PASS);
      System.out.println("Connected to the database successfully");
      System.out.println("Deleting the previous records from the
tables...");
      statement = connection.createStatement();

      String sql = "DELETE FROM locations";
      statement.executeUpdate(sql);
      sql = "DELETE FROM theaters";
      statement.executeUpdate(sql);
      sql = "DELETE FROM showings";
      statement.executeUpdate(sql);
      sql = "DELETE FROM movies";
      statement.executeUpdate(sql);
      System.out.println("The previous records have been successfully
deleted");
      System.out.println("Inserting new records into the tables...");
      for (int i=0; i<movies.length; i++) {
         sql = "INSERT INTO movies (OriginalTitle, ReleaseYear,
TranslatedTitle, Genre, Duration, SuitableUrl, SuitableMessage,
Premiere, Rating, PosterUrl, Plot) " +
         "VALUES ('"+movies[i][1]+"', '"+movies[i][3]+"',
'"+movies[i][2]+"', '"+movies[i][4]+"', '"+movies[i][5]+"',
'"+movies[i][6]+"', '"+movies[i][7]+"', '"+movies[i][9]+"',
'"+movies[i][10]+"', '"+movies[i][8]+"', '"+movies[i][11]+"')";
         statement.executeUpdate(sql);
      }
      for (int i=0; i<locations.length; i++) {
```

```
            sql = "INSERT INTO locations (Name, LastUpdated) VALUES
('"+locations[i][1]+"', '"+locations[i][2]+"')";
            statement.executeUpdate(sql);
        }
        for (int i=0; i<theaters.length; i++) {
            sql = "INSERT INTO theaters (Name, AddressAndPhoneNumber,
LocationName) VALUES ('"+theaters[i][0]+"', '"+theaters[i][1]+"',
'"+theaters[i][2]+"')";
            statement.executeUpdate(sql);
        }
        for (int i=0; i<showings.length; i++) {
            sql = "INSERT INTO showings (MovieOriginalTitle,
MovieReleaseYear, TheaterName, AddressAndPhoneNumber, Hours) " +
            "VALUES ('"+showings[i][0]+"', '"+showings[i][1]+"',
'"+showings[i][2]+"', '"+showings[i][3]+"', '"+showings[i][4]+"')";
            statement.executeUpdate(sql);
        }
        System.out.println("The new records have been successfully
inserted");
        statement.close();
        connection.close();

    }
}
```

Web API

Το web api αποτελείται από τα ακόλουθα PHP scripts τα οποία βρίσκονται στον αετό:

credentials.php. Στην πρώτη οθόνη της εφαρμογής Movie Showtimes ο χρήστης πρέπει να εισάγει το όνομα χρήστη και έναν κλειδάριθμο, ο οποίος (υποτίθεται ότι) αποστέλλεται στον χρήστη αφού αγοράσει την εφαρμογή. Για λόγους ασφαλείας ο κλειδάριθμος αποθηκεύεται κρυπτογραφημένος στη βάση (στον πίνακα credentials) χρησιμοποιώντας τον αλγόριθμο Blowfish. Το script χρησιμοποιεί prepared statements για προστασία από επιθέσεις τύπου SQL injection. Επιπλέον, τα διαπιστευτήρια διαχείρισης βρίσκονται σε ξεχωριστό αρχείο το οποίο είναι τοποθετημένο σε φάκελο πάνω από το web root (το public_html για τον αετό). Στο script περιλαμβάνεται το ακόλουθο τμήμα κώδικα:

```
if ($hash == null) print("Invalid username");
else if (crypt($activationKey,$hash) != $hash) print("Invalid
activation key");
else if (crypt($activationKey,$hash) == $hash) print("Successful
login");
```

Στο σημείο αυτό κρυπτογραφείται εκ νέου ο κλειδάριθμος χρησιμοποιώντας για salt το hash (τον κρυπτογραφημένο κλειδάριθμο). Η συνάρτηση crypt είναι φτιαγμένη έτσι ώστε αν δοθεί ο σωστός κλειδάριθμος και σαν salt το hash, να επιστρέφεται το hash. Με αυτόν τον τρόπο μπορούμε να ελέγξουμε αν έχει εισαχθεί ο σωστός κλειδάριθμος από τον χρήστη, εφόσον δεν είναι δυνατόν να αποκρυπτογραφηθεί το hash έτσι ώστε να κάνουμε με αυτόν τον τρόπο την διασταύρωση.

locations.php. Επιστρέφει τα περιεχόμενα του πίνακα Locations κωδικοποιημένα σε json array. Είναι το μόνο script που δεν χρησιμοποιεί prepared statements διότι δεν υπάρχουν παράμετροι στο sql ερώτημα του script.

showings.php. Επιστρέφει ένα json array με τις ακόλουθες πληροφορίες: τον πρωτότυπο τίτλο της ταινίας, τον τίτλο στα ελληνικά, το έτος κυκλοφορίας, το όνομα της αίθουσας στην οποία προβάλλεται η ταινία, την διεύθυνση και το τηλέφωνο της αίθουσας, τις ημέρες και ώρες προβολών.

movies.php. Το script είναι το ακόλουθο:

```
<?php

include("../db_settings.php");

$originalTitle = $_POST['originalTitle'];
$releaseYear = $_POST['releaseYear'];

$db = new
mysqli("sql3.freesqldatabase.com",$db_user,$db_pass,"sql346520");
$db->query("set names 'utf8'");
$stmt = $db->prepare("SELECT * FROM movies m WHERE m.OriginalTitle =
? AND m.ReleaseYear = ?");
$stmt->bind_param('ss',$originalTitle,$releaseYear);
$stmt->execute();
$stmt-
>bind_result($OriginalTitle,$ReleaseYear,$TranslatedTitle,$Genre,$Dur
ation,$SuitableUrl,$SuitableMessage,$Premiere,$Rating,$PosterUrl,$Plo
t);

$i = 0;

while ($stmt->fetch())
$showings[$i++]=array("OriginalTitle"=>$OriginalTitle,"ReleaseYear"=>
$ReleaseYear,"TranslatedTitle"=>$TranslatedTitle,"Genre"=>$Genre,"Dur
ation"=>$Duration,"SuitableUrl"=>$SuitableUrl,"SuitableMessage"=>$Sui
tableMessage,"Premiere"=>$Premiere,"Rating"=>$Rating,"PosterUrl"=>$Po
sterUrl,"Plot"=>$Plot);
print(json_encode($showings));

$stmt->close();

?>
```

Όπως και τα προηγούμενα scripts χρησιμοποιεί prepared statements. Ο πίνακας που επιστρέφει το sql ερώτημα αποθηκεύεται σε ένα συσχετιστικό πίνακα (association array) δύο διαστάσεων ο οποίος ακολούθως κωδικοποιείται σε ένα json array το οποίο περιέχει τα περιεχόμενα του πίνακα movies.

Movie Showtimes

Η εφαρμογή Movie Showtimes αναπτύχθηκε στο Eclipse. Περιλαμβάνει τέσσερα Activities (Δραστηριότητες) καθένα από τα οποία χρησιμοποιεί xml αρχεία για τη δημιουργία των διεπαφών χρήστη. Στην πρώτη οθόνη της εφαρμογής ο χρήστης πρέπει να εισάγει το όνομα χρήστη και τον κλειδάριθμο που (υποτίθεται ότι) του έχει σταλεί προκειμένου να ενεργοποιήσει την εφαρμογή και να μπορέσει να προχωρήσει στην επόμενη οθόνη. Στην δεύτερη οθόνη ο χρήστης μπορεί να επιλέξει την πόλη στην οποία βρίσκεται και να μεταβεί πατώντας ένα κουμπί στην τρίτη οθόνη όπου εμφανίζεται το πρόγραμμα προβολών για αυτήν την πόλη. Επιλέγοντας κάποια από τις προβολές ο χρήστης περνάει στην τελευταία οθόνη η οποία εμφανίζει κάποιες πληροφορίες για την ταινία.

AndroidManifest.xml

Στο αρχείο AndroidManifest της εφαρμογής αναφέρονται μεταξύ άλλων τα ακόλουθα:

- Τα δικαιώματα τα οποία πρέπει να έχει η εφαρμογή. Η παρούσα εφαρμογή πρέπει να έχει πρόσβαση στο Internet οπότε αυτό δηλώνεται με το tag uses-permission:

```
<uses-permission android:name="android.permission.INTERNET" />
```

- Η ελάχιστη έκδοση του Android API (το API level) στην οποία μπορεί να τρέξει η εφαρμογή. Αναφέρεται επιπλέον, η έκδοση του API στην οποία στοχεύει η εφαρμογή και στην οποία βέβαια έχει δοκιμαστεί.

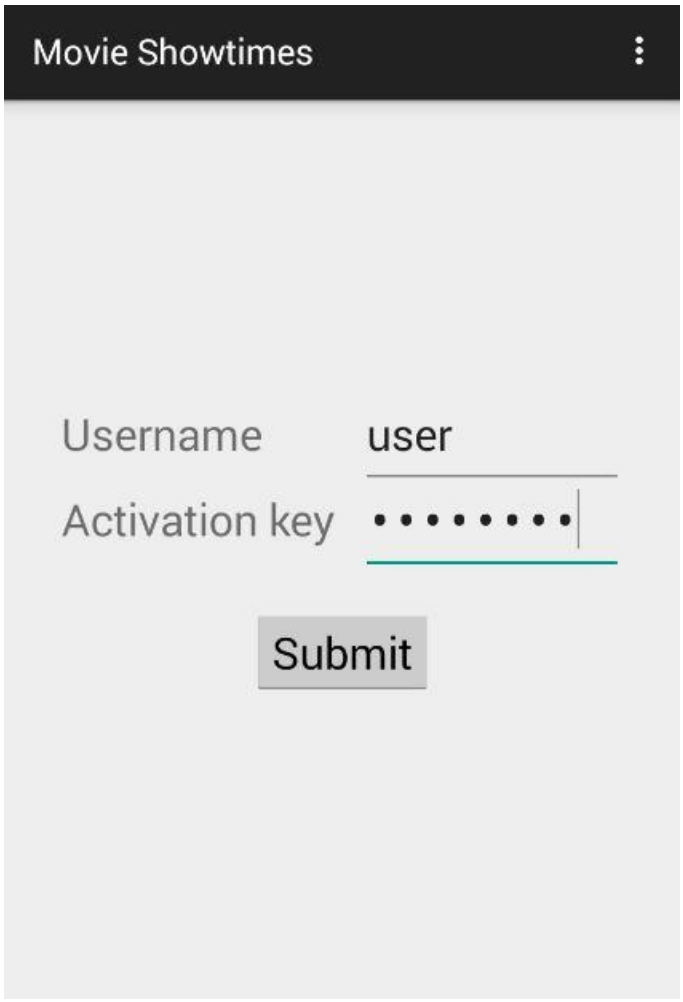
```
<uses-sdk android:minSdkVersion="7" android:targetSdkVersion="21" />
```

  Αν η εφαρμογή εγκατασταθεί σε συσκευή η οποία διαθέτει μεγαλύτερο API level από αυτό που αναγράφεται στο targetSdkVersion, το σύστημα μπορεί να ενεργοποιήσει συμπεριφορές συμβατότητας (compatibility behaviors) προκειμένου να τρέξει η εφαρμογή.
- Το όνομα και το εικονίδια της εφαρμογής. Τα εικονίδια βρίσκονται στους φακέλους drawable. Το Eclipse διαθέτει εργαλείο για την δημιουργία εικονιδίων τα οποία είναι κατάλληλα σε οθόνες με διαφορετική πυκνότητα pixel.
- Τα Activities τα οποία περιέχει η εφαρμογή και το πώς συνδέονται μεταξύ τους. Για παράδειγμα το ShowingsActivity με το οποίο εμφανίζεται το πρόγραμμα των προβολών στην τρίτη οθόνη, αναφέρει σαν μητρικό στοιχείο το LocationsActivity:

```
android:parentActivityName="com.gmoutzur.movies.LocationsActivity"
```

LoginActivity

Το πρώτο Activity είναι το LoginActivity.



Η διεπαφή χρήστη του LoginActivity

Όπως και τα υπόλοιπα Activities της εφαρμογής, επεκτείνει την κλάση ActionBarActivity, οπότε με αυτό τον τρόπο προστίθεται η Action bar (μπάρα ενεργειών) στο πάνω μέρος την οθόνης. Για την δημιουργία μενού ενεργειών στην Action bar περιλαμβάνει τις μεθόδους `onCreateOptionsMenu()` και `onOptionsItemSelected()`. Γενικά, η πρώτη μέθοδος κάνει inflate (δημιουργεί δηλαδή αντικείμενα τα οποία περιγράφονται σε αρχεία xml) κάποιο αρχείο xml το οποίο βρίσκεται στον φάκελο menu των resources. Το αρχείο αυτό περιλαμβάνει τα στοιχεία του μενού. Στο αρχείο login.xml περιλαμβάνεται ένα στοιχείο μενού:

```
<item android:id="@+id/action_settings" android:orderInCategory="100"
      android:title="@string/action_settings"
      app:showAsAction="never"/>
```

Αν ο χρήστης κάνει κλικ σε αυτό το στοιχείο μεταβαίνει στις ρυθμίσεις του συστήματος. Ο χειρισμός αυτών των κλικ γίνεται στη συνάρτηση `onOptionsItemSelected()`.

Η μέθοδος onCreate() καλείται γενικά όταν δημιουργείται κάποιο αντικείμενο Activity. Στην μέθοδο onCreate() καλείται η μέθοδος setContentView(),

```
setContentView(R.layout.activity_login);
```

η οποία στην προκειμένη περίπτωση παίρνει σαν παράμετρο το αρχείο activity_login.xml στο οποίο περιλαμβάνονται τα στοιχεία της διεπαφής χρήστη για αυτό το Activity. Η διεπαφή χρήστη για αυτήν την οθόνη περιλαμβάνει συνολικά πέντε στοιχεία για την διάταξη των οποίων χρησιμοποιήθηκε το RelativeLayout:

- Δύο TextViews (τα TextViews χρησιμοποιούνται για την εμφάνιση κειμένου), τα username και activation_key.
- Δύο EditTexts (τα EditTexts χρησιμοποιούνται για την εισαγωγή κειμένου), τα username_edit και activation_key_edit.
- Το κουμπί credentials_button.

Όταν πατηθεί το κουμπί καλείται η συνάρτηση checkCredentials() όπου γίνεται η επαλήθευση των διαπιστευτηρίων που έχει εισάγει ο χρήστης. Στην συνάρτηση περιλαμβάνεται η εντολή:

```
new CheckCredentials().execute(username,activationKey);
```

Με αυτήν την εντολή δημιουργείται ένα στιγμιότυπο της κλάσης CheckCredentials η οποία επεκτείνει την ειδική κλάση AsyncTask. Με την AsyncTask μπορούμε να δημιουργήσουμε ένα παράλληλο νήμα. Έτσι, η επεξεργασία για την επαλήθευση των διαπιστευτηρίων *δεν* γίνεται στο νήμα διεπαφής χρήστη (UI thread), προκειμένου η διεπαφή να έχει καλή απόκριση. Η AsyncTask χρησιμοποιεί generics:

```
private class CheckCredentials extends AsyncTask<String, Void, String>
```

Η πρώτη παράμετρος αφορά τον τύπο των δεδομένων που εισάγονται για επεξεργασία, ενώ η τελευταία τον τύπο των δεδομένων που επιστρέφονται τελικά. Η AsyncTask περιλαμβάνει τις εξής μεθόδους:

- onPreExecute(). Καλείται στο νήμα διεπαφής χρήστη πριν εκτελεστεί η εργασία. Στην προκειμένη περίπτωση δημιουργείται ένα progress dialog με τις εντολές

  ```
  progressDialog.setMessage(getString(R.string.checking_credentials));
  progressDialog.show();
  ```

  το οποίο παραμένει στην οθόνη μέχρι να ολοκληρωθεί η επεξεργασία.
- doInBackground(). Καλείται στο παράλληλο νήμα μετά την onPreExecute(). Το αποτέλεσμα της επεξεργασίας περνιέται στην επόμενη μέθοδο. Εδώ, αποστέλλονται στο web script credentials.php, το οποίο βρίσκεται στον aetos, τα διαπιστευτήρια που εισήγαγε ο χρήστης. Τα διαπιστευτήρια αποστέλλονται κρυπτογραφημένα για λόγους ασφάλειας. Το script ελέγχει αν αυτά υπάρχουν στην βάση και επιστρέφει το κατάλληλο μήνυμα. Να σημειωθεί ότι για τις ανάγκες της εργασίας έχουν εισαχθεί στον πίνακα credentials τα «user» και «activate» (το τελευταίο, επίσης κρυπτογραφημένο

όπως αναφέρθηκε σε προηγούμενη ενότητα), τα οποία μπορούν να εισαχθούν για να γίνει η μετάβαση στην επόμενη οθόνη.
Η κρυπτογράφηση των διαπιστευτηρίων γίνεται με χρήση SSL. Η αρχή πιστοποίησης του πιστοποιητικού του aetos δεν θεωρείται όμως αξιόπιστη από το Android. Έτσι, σε πρώτη φάση, το πιστοποιητικό έπρεπε να αποθηκευτεί σε ένα keystore με φορμά .bks στον φάκελο raw των resources της εφαρμογής. Για τη δημιουργία του keystore χρησιμοποιήθηκε το εργαλείο Portecle. Από εκεί και πέρα, η εφαρμογή χρησιμοποιεί την βοηθητική κλάση MyHttpClient. Η κλάση MyHttpClient επεκτείνει την DefaultHttpClient και κάνει override την μέθοδο createClientConnectionManager(). Σε αυτήν την μέθοδο περιλαμβάνεται η ακόλουθη εντολή

```
registry.register(new Scheme("https", newSslSocketFactory(), 443));
```

η οποία δηλώνει ότι η μέθοδος newSslSocketFactory() θα χρησιμοποιηθεί για την δημιουργία ενός SSLSocketFactory. Το SSLSocketFactory είναι υπεύθυνο για την έλεγχο του πιστοποιητικού. Η μέθοδος newSslSocketFactory() περιλαμβάνει την ακόλουθη εντολή

```
SSLSocketFactory sf = new SSLSocketFactory(trusted);
```

με την οποία το keystore tei.bks περνιέται σαν παράμετρος στο SSLSocketFactory. Έτσι, ο έλεγχος που γίνεται για την αξιοπιστία του πιστοποιητικού είναι επιτυχής.

- onPostExecute(). Καλείται στο νήμα διεπαφής χρήστη όταν ολοκληρωθεί η επεξεργασία. Εδώ, περνιέται στην onPostExecute() το μήνυμα το οποίο επέστρεψε το credentials.php. Αν δεν υπάρχει το όνομα χρήστη ή ο κλειδάριθμος, εμφανίζεται στην οθόνη ένα toast το οποίο ενημερώνει τον χρήστη κατάλληλα. Αν η επαλήθευση των διαπιστευτηρίων γίνει επιτυχώς, γίνεται η μετάβαση στην επόμενη οθόνη με τις εντολές:

```
Intent intent = new Intent(LoginActivity.this, LocationsActivity.class);
startActivity(intent);
```

Η (επιτυχής) εισαγωγή των διαπιστευτηρίων πρέπει να γίνεται όμως μόνο μία φορά για όσο είναι εγκατεστημένη η εφαρμογή. Η εφαρμογή πρέπει να «θυμάται» ότι έχουν εισαχθεί επιτυχώς. Έτσι, με τις ακόλουθες εντολές

```
SharedPreferences sharedPref =
(LoginActivity.this).getPreferences(Context.MODE_PRIVATE);
SharedPreferences.Editor editor = sharedPref.edit();
editor.putBoolean("Registered",true);
editor.commit();
```

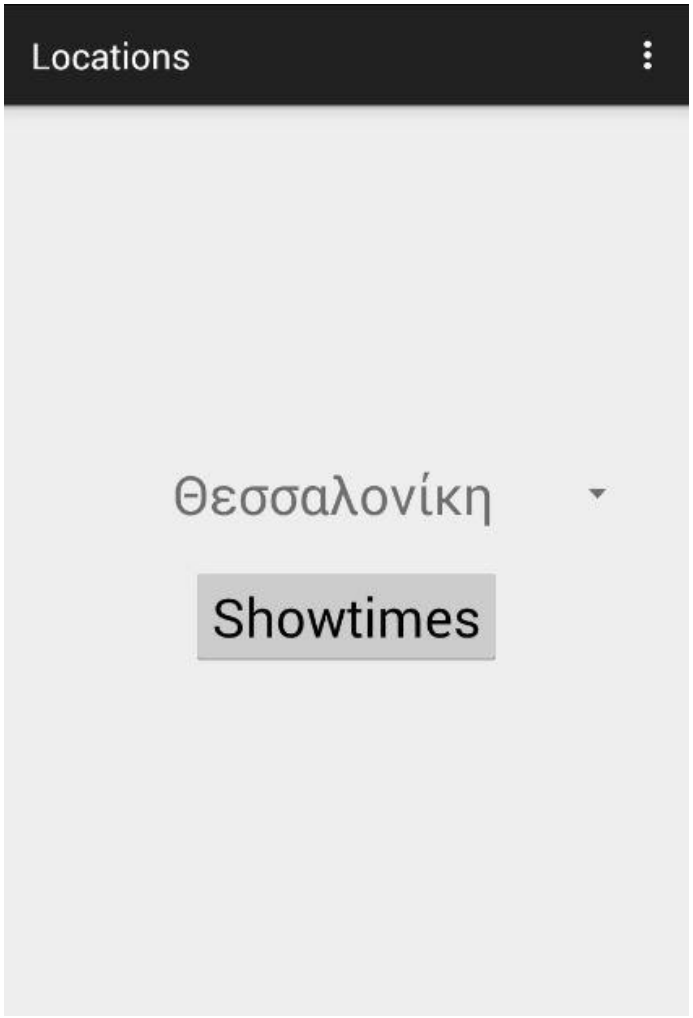
η εφαρμογή αποθηκεύει σε ένα αρχείο μία μεταβλητή Boolean με την τιμή true. Όταν αποθηκεύουμε σε αρχείο χρησιμοποιώντας την κλάση SharedPreferences, μπορούμε να αποθηκεύσουμε μόνο ζευγάρια κλειδιών-τιμών (key-values). Στην μέθοδο onCreate() γίνεται ανάγνωση της τιμής και αν η τιμή είναι true γίνεται κατευθείαν μετάβαση στην επόμενη οθόνη με τις ακόλουθες εντολές:

```
SharedPreferences sharedPref =
this.getPreferences(Context.MODE_PRIVATE);
boolean isRegister = sharedPref.getBoolean("Registered",false);

if (isRegister) {
    Intent intent = new Intent(this, LocationsActivity.class);
    startActivity(intent);
}
```

Επιπλέον, αφού γίνει η μετάβαση στην επόμενη οθόνη, θα πρέπει ο χρήστης πατώντας το back να φεύγει από την εφαρμογή και όχι να εμφανίζεται η πρώτη οθόνη της εφαρμογής. Για να μην προστίθεται το LoginActivity στο backstack (στοίβα που περιέχει τα Activities στα οποία μεταβαίνει ο χρήστης πατώντας το back), έχει προστεθεί στο μανιφέστο το `android:noHistory="true"` για αυτό το Activity.

LocationsActivity



Η διεπαφή χρήστη του LocationsActivity

Στην μέθοδο onCreate() περιλαμβάνονται οι ακόλουθες εντολές:

```
setContentView(R.layout.activity_locations);
new FetchLocations().execute();
```

Το activity_locations.xml περιλαμβάνει τα στοιχεία της διεπαφής χρήστη, τα οποία για να τα διατάξει χρησιμοποιεί το LinearLayout. Η διεπαφή χρήστη για αυτό το Activity περιλαμβάνει ένα Spinner (drop-down list) και ένα κουμπί για την μετάβαση στην επόμενη οθόνη. Μέσω του Spinner ο χρήστης μπορεί να επιλέξει την πόλη για την οποία επιθυμεί να δει το πρόγραμμα των ταινιών.

Εφόσον το Activity διαθέτει Spinner, απαιτείται η υλοποίηση του interface OnItemSelectedListener, το οποίο περιλαμβάνει τις μεθόδους onItemSelected() και onNothingSelected().

Η κλάση FetchLocations επεκτείνει την κλάση AsyncTask, έτσι ώστε η επεξεργασία να γίνει σε ένα νέο νήμα. Στην προκειμένη περίπτωση η AsyncTask δεν παίρνει κάποια δεδομένα σαν είσοδο.

- doInBackground(): Στην μέθοδο doInBackground() της FetchLocations περιλαμβάνονται οι εντολές:

  ```
  JSONParser jParser = new JSONParser(LocationsActivity.this);
  return jParser.getLocations();
  ```

  Εδώ, χρησιμοποιείται η μέθοδος getLocations() της βοηθητικής κλάσης JSONParser για την μετατροπή του JSON array το οποίο αποστέλλει το web script locations.php σε String.

- onPostExecute(): Η μέθοδος onPostExecute() παίρνει σαν παράμετρο το String που επέστρεψε η doInBackground(). Εντός της onPostExecute() γίνονται τα ακόλουθα: Δημιουργείται ένα αντικείμενο JSONArray χρησιμοποιώντας το προαναφερθέν String, ένα for loop περνάει από όλα τα αντικείμενα τύπου JSONObject τα οποία περιέχει το JSONArray, από τα αντικείμενα JSONObject παίρνουμε τελικά τα περιεχόμενα του πίνακα Locations της βάσης, δηλαδή το όνομα της πόλης και την ημερομηνία που έγινε η ενημέρωση των δεδομένων για αυτήν την πόλη. Το τελευταίο βήμα γίνεται με τις ακόλουθες εντολές:

  ```
  location = jsonObject.getString("Name");
  lastUpdated = jsonObject.getString("LastUpdated");
  ```

  Τα strings location και lastUpdated εισάγονται στις ArrayLists locationsList και lastUpdatedList με τις εντολές

  ```
  locationsList.add(location);
  lastUpdatedList.add(lastUpdated);
  ```

  Τέλος, η onPostExecute() καλεί την μέθοδο setSpinner().

Με την setSpinner, το Spinner συνδέεται με τον αντάπτορα ArrayAdapter, ο οποίος παίρνει σαν παράμετρο την locationsList που περιλαμβάνει τις πόλεις. Έτσι, οι πόλεις που είναι αποθηκευμένες σε Strings μετατρέπονται τελικά σε TextViews για την τοποθέτησή τους στο Spinner. Επιπλέον, ο ArrayAdapter παίρνει σαν παράμετρο το αρχείο spinner_item.xml, το οποίο καθορίζει το layout των TextViews του Spinner. Στο αρχείο περιλαμβάνονται οι εξής γραμμές

```
android:gravity="center"
android:textSize="30sp"
```

με τις οποίες τα περιεχόμενα των TextViews εμφανίζονται κεντραρισμένα, σε μέγεθος 30sp.

Όταν ο χρήστης πατήσει το κουμπί καλείται η μέθοδος displayShowings(). Σε αυτήν δημιουργείται ένα αντικείμενο Intent με το οποίο γίνεται η επικοινωνία με το ShowingsActivity. Το Intent μεταφέρει επίσης το όνομα της πόλης καθώς και την ημερομηνία που έγινε η ανανέωση των δεδομένων.

ShowingsActivity

Η διεπαφή χρήστη του ShowingsActivity

Το ShowingsActivity χρησιμοποιεί μία ListView για να εμφανίσει το πρόγραμμα των προβολών για μία πόλη. Εφόσον χρησιμοποιείται η ListView, υλοποιείται το interface OnItemClickListener, το οποίο περιέχει μόνο την μέθοδο onItemClick() η οποία καλείται όταν ο χρήστης κάνει κλικ σε κάποιο στοιχείο της ListView.

Στην μέθοδο onCreate() περιλαμβάνονται οι εντολές

```
Intent intent = getIntent();
```

```
location = intent.getStringExtra(LocationsActivity.EXTRA_LOCATION);
lastUpdated  =  intent.getStringExtra(LocationsActivity.EXTRA_LASTUPDATED);
new FetchShowings().execute(location);
```

με τις οποίες αποθηκεύονται στα Strings location και lastUpdated τα δεδομένα που μετέφερε το αντικείμενο Intent στο ShowingsActivity.

Η FetchShowings επεκτείνει την κλάση AsyncTask, παίρνοντας σαν είσοδο με τη μορφή String την τοποθεσία που επέλεξε ο χρήστης.

- doInBackground(): Στην μέθοδο doInBackground() περιλαμβάνονται οι εντολές:

  ```
  JSONParser jParser = new JSONParser(ShowingsActivity.this);
  return jParser.getShowings(strings[0]);
  ```

  Εδώ, χρησιμοποιείται η μέθοδος getShowings() της κλάσης JSONParser για την μετατροπή του JSON array το οποίο αποστέλλει το web script showings.php σε String. Η getShowings παίρνει σαν παράμετρο την πόλη, η οποία στέλνεται στο showings.php.

- onPostExecute(): Η μέθοδος onPostExecute() παίρνει σαν παράμετρο το String που επέστρεψε η doInBackground(). Με τρόπο ανάλογο με αυτόν που χρησιμοποιήθηκε στην onPostExecute() του LocationsActivity(), μπορούμε να πάρουμε τα δεδομένα που χρειαζόμαστε από το String. Τα δεδομένα εισάγονται τελικά σε μία ArrayList και καλείται η μέθοδος setListView().

Στην setListView() δημιουργείται ο αντάπτορας CustomSimpleAdapter μέσω του οποίου συνδέονται τα δεδομένα που υπάρχουν στην ArrayList showingsList με την ListView. Ο αντάπτορας παίρνει σαν παραμέτρους: την showingsList, το αρχείο list_item.xml το οποίο περιλαμβάνει το layout για την κάθε γραμμή της ListView, και άλλους δύο πίνακες με τους οποίους γίνεται αντιστοίχιση των πεδίων δεδομένων της showingsList με τα TextViews που περιλαμβάνονται στο αρχείο list_item.xml. Επιπλέον, εισάγεται ένας header στην ListView ο οποίος αναφέρει την ημερομηνία που έγινε η ανανέωση των δεδομένων για αυτήν την πόλη.

Είναι σημαντικό να σημειωθεί ότι μπορεί για κάποια προβολή να μην εμφανίζονται όλες οι πληροφορίες. Για παράδειγμα, για ορισμένες ταινίες δεν υπάρχει μεταφρασμένος τίτλος. Το πρόβλημα σε αυτή την περίπτωση είναι ότι το ίδιο layout (ένα κενό layout σε αυτή την περίπτωση) το οποίο χρησιμοποιεί η ListView για να εμφανίσει κάποιο στοιχείο (π.χ. τον μεταφρασμένο τίτλο όπως αναφέρθηκε πριν), χρησιμοποιείται εκ νέου για λόγους αποδοτικότητας για την εμφάνιση των αντίστοιχων στοιχείων. Έτσι, όσο μετακινείται ο χρήστης στην ListView εξαφανίζονται σταδιακά όλα τα αντίστοιχα στοιχεία! Για τον λόγο αυτό, μπορούμε να δημιουργήσουμε έναν custom αντάπτορα, όπου κάνουμε override την μέθοδο getView. Στην μέθοδο getView του CustomSimpleAdapter περιλαμβάνονται εντολές όπως οι ακόλουθες

```
if (translatedTitle.length() == 0)
textViewTranslatedTitle.setVisibility(View.GONE);
else textViewTranslatedTitle.setVisibility(View.VISIBLE);
```

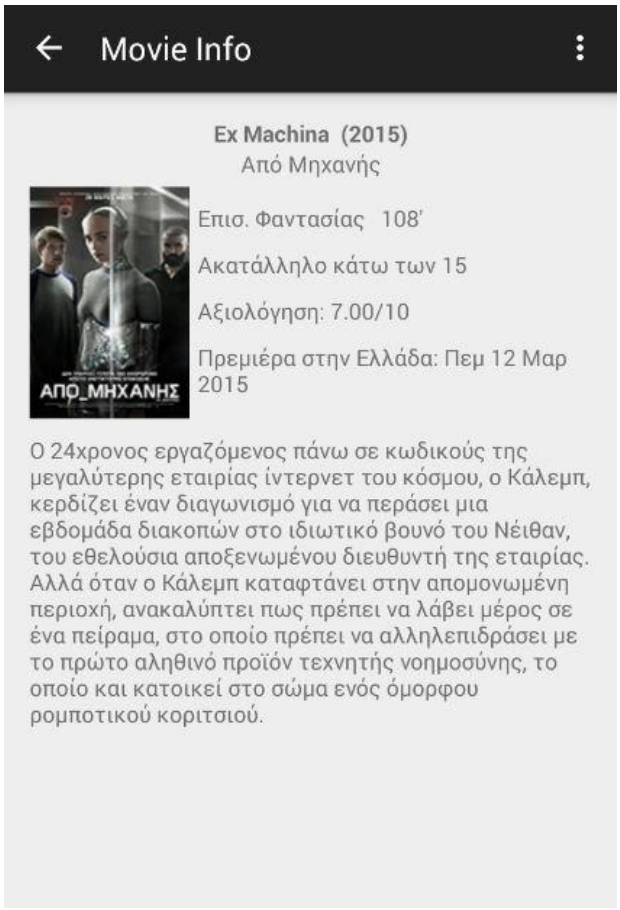
όπου γίνεται ένας έλεγχος για να διαπιστωθεί αν υπάρχει κενό String, οπότε τίθεται το visibility σε gone και το αντίστοιχο TextView δεν εμφανίζεται. Στην αντίθετη περίπτωση το TextView εμφανίζεται στην οθόνη.

Το ShowingsActivity περιλαμβάνει επίσης τη μέθοδο onSaveInstanceState() όπου αποθηκεύεται σε ένα αντικείμενο Bundle η θέση του πρώτου ορατού στοιχείου της ListView. Σε περίπτωση που ο χρήστης αλλάξει τον προσανατολισμό (orientation) της συσκευής, το Activity καταστρέφεται και επαναδημιουργείται, οπότε πρέπει να επαναφέρουμε την ListView στην προηγούμενη κατάσταση. Στην μέθοδο onCreate() γίνεται έλεγχος για το αν το αντικείμενο Bundle περιέχει πληροφορίες, και στην περίπτωση που το Bundle δεν είναι κενό, αποθηκεύεται η θέση της ListView στην μεταβλητή listIndex. Στην μέθοδο setListView που αναφέρθηκε προηγουμένως επαναφέρουμε την ListView στην προηγούμενη θέση με την εντολή

```
listView.setSelection(listIndex);
```

Στην μέθοδο onItemClick() δημιουργείται ένα αντικείμενο Intent το οποίο στέλνεται στο MoviesActivity. Στο Intent προστίθεται ο πρωτότυπος τίτλος της ταινίας και το έτος κυκλοφορίας, προκειμένου να εμφανιστούν στην επόμενη οθόνη πληροφορίες για αυτήν την ταινία.

MoviesActivity



Η διεπαφή χρήστη του MoviesActivity

Στην μέθοδο onCreate() υπάρχει η εντολή

```
setContentView(R.layout.activity_movies);
```

Το activity_movies.xml περιλαμβάνει τα στοιχεία της διεπαφής χρήστη, τα οποία για να τα διατάξει χρησιμοποιεί RelativeLayout. Επιπλέον, η onCreate() περιλαμβάνει τις ακόλουθες εντολές

```
Intent intent = getIntent();
originalTitle=intent.getStringExtra(ShowingsActivity.EXTRA_ORIGINALTITLE);
releaseYear = intent.getStringExtra(ShowingsActivity.EXTRA_RELEASEYEAR);
new FetchMovies().execute(originalTitle, releaseYear);
```

με τις οποίες αποθηκεύονται στα Strings originalTitle και releaseYear τα δεδομένα που μετέφερε το αντικείμενο Intent στο MoviesActivity.

Η FetchMovies επεκτείνει την κλάση AsyncTask, παίρνοντας σαν είσοδο με τη μορφή String τον πρωτότυπο τίτλο και το έτος κυκλοφορίας της ταινίας.

- doInBackground(): Στην μέθοδο doInBackground() περιλαμβάνονται οι εντολές:

  ```
  JSONParser jParser = new JSONParser(MoviesActivity.this);
  return jParser.getMovies(strings[0], strings[1]);
  ```

  Χρησιμοποιείται η μέθοδος getMovies() της κλάσης JSONParser για την μετατροπή του JSON array το οποίο αποστέλλει το web script movies.php σε String. Η getMovies παίρνει σαν παραμέτρους τα δεδομένα που μετέφερε το Intent, τα οποία στέλνονται στο movies.php.

- onPostExecute(): Η μέθοδος onPostExecute() παίρνει σαν παράμετρο το String που επέστρεψε η doInBackground(). Με τρόπο ανάλογο με αυτόν που χρησιμοποιήθηκε στην onPostExecute() προηγουμένως, μπορούμε να πάρουμε τα δεδομένα που χρειαζόμαστε από το String. Τα δεδομένα εισάγονται σε διάφορα Strings (όπως το translatedTitle) και καλείται η μέθοδος setViews().

Στην μέθοδο setViews() με εντολές όπως οι ακόλουθες

```
TextView textViewTranslatedTitle = (TextView)
findViewById(R.id.translatedTitle);
if (translatedTitle.length() == 0)
textViewTranslatedTitle.setVisibility(View.GONE);
else textViewTranslatedTitle.setText(translatedTitle);
```

γίνεται ένας έλεγχος για να διαπιστωθεί αν υπάρχουν κενά Strings, οπότε σε αυτήν την περίπτωση το visibility του View τίθεται σε gone και δεν εμφανίζεται, ούτε και καταλαμβάνει χώρο στην οθόνη. Επιπλέον, για την εμφάνιση της αφίσας της ταινίας υπάρχουν οι εντολές

```
ImageView poster = (ImageView) findViewById(R.id.poster);
new DownloadImage(poster).execute(posterUrl);
```

προκειμένου το κατέβασμα της εικόνας να γίνει παρασκηνιακά, σε ένα νέο νήμα, για την δημιουργία του οποίου χρησιμοποιείται η κλάση DownloadImage

η οποία επεκτείνει την AsyncTask. Σαν είσοδο στην AsyncTask περνιέται το URL της αφίσας.

Λοιπές παρατηρήσεις

Στο αρχείο strings.xml του φακέλου values-el, υπάρχουν μεταφράσεις για όλες τις πληροφορίες που εμφανίζονται στα αγγλικά. Έτσι, αν η γλώσσα του συστήματος είναι τα ελληνικά, το σύστημα θα χρησιμοποιήσει τα Strings αυτού του αρχείου αντί για τα Strings που βρίσκονται στο strings.xml του φακέλου values.

Επιπλέον, στο φάκελο layout-large υπάρχουν διάφορα αρχεία xml τα οποία θα χρησιμοποιηθούν από το σύστημα για τη δημιουργία των διεπαφών χρήστη αν η οθόνη είναι μεγάλη. Διαφορετικά θα χρησιμοποιηθούν τα αρχεία xml που βρίσκονται στον φάκελο layout των resources της εφαρμογής.

Πηγές

Γενικά για το Android:

http://www.tutorialspoint.com/android/android_overview.htm

Android NDK:

https://developer.android.com/tools/sdk/ndk/index.html

AndroidManifest.xml

http://developer.android.com/guide/topics/manifest/manifest-intro.html

Android Developer Tools:

http://developer.android.com/tools/help/adt.html

Android Studio:

http://developer.android.com/training/basics/firstapp/running-app.html

Android Runtime:

https://source.android.com/devices/tech/dalvik/index.html

http://www.techotopia.com/index.php/An_Overview_of_the_Android_Architecture

http://anandtech.com/show/8231/a-closer-look-at-android-runtime-art-in-android-l/

Application Components:

http://developer.android.com/guide/components/fundamentals.html

https://source.android.com/devices/tech/security/overview/app-security.html

Learning Android, O'Reilly (Marko Gargenta and Masumi Nakamura)

Διεπαφή Χρήστη:

http://developer.android.com/guide/topics/ui/overview.html

Γραφικά και Animations:

http://developer.android.com/guide/topics/graphics/index.html

Hardware Accelerated 2D Rendering for Android, Feb 19, 2013 / Android Builders Summit (Jim Huang)

Ασφάλεια:

http://source.android.com/devices/tech/security/overview/index.html

Δοκιμή:

http://www.vogella.com/tutorials/AndroidTesting/article.html

https://developer.android.com/tools/testing/testing_android.html

Google Services:
http://androidmobileapps.wikispaces.com/file/detail/Google+Play+Services.potx/529546152

https://developer.android.com/google/index.html

http://developer.android.com/google/play-services/index.html

https://developer.android.com/google/play-services/games.html

https://developer.android.com/google/play-services/location.html

https://developer.android.com/google/play-services/drive.html

https://developer.android.com/google/play/billing/index.html

https://developer.android.com/google/play-services/ads.html

http://developer.android.com/google/play-services/setup.html

http://developer.android.com/sdk/installing/adding-packages.html