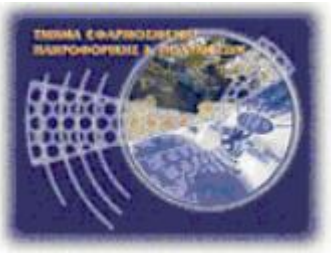
Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Μηχανικών Πληροφορικής



Πτυχιακή εργασία

Ανάπτυξη Android εφαρμογής για την αυτόματη μετάφραση κειμένου που υπάρχει σε εικόνα

Περάκη Αθηνά

Α.Μ. 1973

Επιβλέπων Καθηγητής: Τριανταφυλλίδης Γεώργιος

Σύνοψη

Αυτή η εργασία ασχολείται με την ανάπτυξη μίας εφαρμογής αυτόματης μετάφρασης κειμένου από φωτογραφίες. Συγκεκριμένα αναλύεται η διαδικασία ανάπτυξης μίας εφαρμογής για κινητά με λειτουργικό σύστημα Android με την οποία ο χρήστης μπορεί να τραβήξει μία φωτογραφία ενός κειμένου, η οποία αποστέλλεται σε έναν server που αναλαμβάνει να αναγνωρίσει το κείμενο και να το μεταφράσει. Η μετάφραση κατόπιν αποστέλλεται πίσω στον χρήστη.

Επιπλέον παρουσιάζονται κάποια στοιχεία τόσο για την οπτική αναγνώριση χαρακτήρων (OCR), την αυτόματη μετάφραση κειμένου και γίνεται μία εισαγωγή στις δυνατότητες του Android και των τεχνολογιών Servlet της Java.

Abstract

This thesis deals with the development of an application that automatically translates text from photographs. The project is separated into an Android application that allows the user to take a picture containing text. The image is sent to a remote server that recognizes the characters in it, translates it to another language and sends the text back to the user.

In addition we give a basic introduction to the optical character recognition and automatic translation technologies, as well as the capabilities of the Android operating system and the Java Servlet technology.

Περιεχόμενα

1 Εισαγωγή

1.1 Περίληψη

Αυτή η πτυχιακή εργασία παρουσιάζει την διαδικασία ανάπτυξης μίας εφαρμογής αυτόματης μετάφρασης κειμένου για κινητά Android. Η εφαρμογή ζητάει από τον χρήστη να πάρει μία φωτογραφία ενός ξενόγλωσσου κειμένου, το οποίο αποστέλλει σε έναν απομακρυσμένο server μέσω internet. Αυτός το αναλύει με χρήση τεχνολογίας αναγνώρισης χαρακτήρων και χρησιμοποιεί μία υπηρεσία αυτόματης μετάφρασης για να πάρει την ελληνική μετάφραση του κειμένου και να στην στείλει πίσω την εφαρμογή στο κινητό.

1.2 Κίνητρο για την διεξαγωγή της εργασίας

Τα κινητά τηλέφωνα είναι πλέον ένα μέρος της καθημερινότητας ενός μεγάλου μέρους του πληθυσμού του πλανήτη. Όμως πλέον δεν πρόκειται για χαζές συσκευές που η μοναδική εργασία που επιτελούν είναι να παίρνουν ένα τηλέφωνο, αλλά πρόκειται για μικρούς ηλεκτρονικούς υπολογιστές, που πολλές φορές έχουν περισσότερες δυνατότητες από σταθερούς υπολογιστές παλιότερης γενιάς.

Σε αυτό το πλαίσιο θελήσαμε να δημιουργήσουμε μία εφαρμογή για τέτοια τηλέφωνα που συνδυάζει τις τεχνολογίες της οπτικής αναγνώρισης χαρακτήρων και της αυτόματης μετάφρασης κειμένου. Θεωρώντας πως και οι δύο τομείς δίνουν συναρπαστικές ευκαιρίες στους χρήστες προσπαθήσαμε να εξερευνήσουμε πως μπορούν να συνδυαστούν σε μία σύγχρονη εφαρμογή.

1.3 Σκοπός και στόχοι εργασίας

Σκοπός της εργασίας είναι η ανάπτυξη μίας εφαρμογής για κινητά τηλέφωνα με λειτουργικό σύστημα Android μέσω της οποίας ο χρήσης θα μπορεί να τραβάει μία φωτογραφία ενός ξενόγλωσσου κειμένου που τον ενδιαφέρει. Η εφαρμογή θα αναλαμβάνει να αναγνωρίσει το κείμενο που περιέχει και να το μεταφράσει αυτόματα στα ελληνικά, χωρίς άλλη παρέμβαση. Έπειτα θα εμφανίζει το κείμενο του αποτελέσματος.

Μέσω της παραπάνω διαδικασίας θέλουμε να δούμε την κατάσταση που θα αντιμετωπίσει ένας προγραμματιστής που προσπαθεί να συνδυάσει αυτές τις τεχνολογίες.

1.4 Δομή εργασίας

Κεφάλαιο 1 – Περίληψη: Πρόκειται για αυτό το κεφάλαιο, με γενικές πληροφορίες για την εργασία.

Κεφάλαιο 2 – Έξυπνα τηλέφωνα και το λειτουργικό σύστημα Android: Σε αυτό το κεφάλαιο παρουσιάζουμε κάποια βασικά στοιχεία για τα έξυπνα τηλέφωνα και τα βασικά στοιχεία των κινητών Android.

Κεφάλαιο 3 – Εγκατάσταση εργαλείων: Αυτό το κεφάλαιο δείχνει τα εργαλεία που χρησιμοποιήσαμε και από πού μπορεί να τα βρει ο χρήστης της εφαρμογής.

Κεφάλαιο 4 – Ανάλυση εφαρμογής: Το τέταρτο κεφάλαιο περιέχει την ανάλυση του κώδικα της εφαρμογής και τεχνικές πληροφορίες σχετικά με τις ρυθμίσεις και το πρωτόκολλο επικοινωνίας.

Κεφάλαιο 5 – Οδηγίες χρήσης εφαρμογής: Εδώ δίνουμε τις απαραίτητες οδηγίες ώστε κάποιος να μπορεί να τρέξει την εφαρμογή μας από τα παραδοτέα που προμιθεύουμε.

Κεφάλαιο 6 – Συμπεράσματα: Στο κεφάλαιο αυτό παρουσιάζουμε τα συμπεράσματα που προέκυψαν μέσω της διαδικασίας ανάπτυξης της εφαρμογής.

2 Έξυπνα τηλέφωνα και το λειτουργικό σύστημα Android

2.1 Γενικά

Τα τελευταία χρόνια έχει γίνει μία πολύ μεγάλη αλλαγή στον χώρο της πληροφορικής με την πολύ μεγάλη εξάπλωση των έξυπνων κινητών ή smartphones, τα οποία δίνουν στους χρήστες τους δυνατότητες να κάνουν πράγματα που παλιότερα γίνονταν μόνο με υπολογιστές και κυρίως καθιστούν εφικτή την ανάπτυξη και την χρήση διαφόρων προγραμμάτων που δεν έχουν γραφτεί μόνο από τον κατασκευαστή της εκάστοτε συσκευής όπως παλιότερα.

Αν και στο παρελθόν είχαν γίνει προσπάθειες ανάπτυξης συσκευών που προσέφεραν επιπλέον δυνατότητες, η δυνατότητα για την εμφάνιση αυτών των συσκευών δόθηκε από την εξέλιξη του hardware, το οποίο πλέον είναι συγκρίσιμο με τους συμβατικούς υπολογιστές παλαιότερων χρόνων. Η επιπλέον υπολογιστική ισχύ δίνει τη δυνατότητα να χρησιμοποιηθούν με πολύ διαφορετικό τρόπο από τα κλασσικά τηλέφωνα.

Η βασικότερη αλλαγή έγινε στη νοοτροπία των εταιρειών, οι οποίες πριν από την εμφάνιση αυτών των συσκευών θεωρούσαν ότι τα κινητά τηλέφωνα έπρεπε να είναι απόλυτα αποκλεισμένα από τους χρήστες ώστε να μην μπορούν να κάνουν καμία αλλαγή σε αυτά. Πλέον ενθαρρύνουν και βοηθάνε τους προγραμματιστές να γράφουν νέες εφαρμογές για αυτά προσφέροντας τα απαραίτητα εργαλεία και documentation.

Αυτό οφείλεται στη βασική ιδέα πως οι επιπλέον εφαρμογές δίνουν μεγαλύτερη αξία στη συσκευή και την κάνουν περισσότερο ελκυστική στους καταναλωτές, ενώ ταυτόχρονα δεν χρειάζεται να αναπτύσσουν όλες τις εφαρμογές που μπορεί να χρειαστεί κάποιος, γλυτώνοντας και το κόστος τους.

2.2 Έξυπνα τηλέφωνα

2.2.1 Τι είναι έξυπνο τηλέφωνο

Ένας απόλυτος ορισμός για το τι είναι έξυπνο τηλέφωνο δεν είναι εύκολο να δοθεί, καθώς υπάρχουν πάρα πολλές παραλλαγές και η έννοια του όρου έχει αλλάξει με το πέρασμα του χρόνου. Όμως είναι γενικά αποδεκτό πώς ένα έξυπνο τηλέφωνο είναι μία συσκευή κινητού τηλεφώνου που προσφέρει δυνατότητα σύνδεσης στο διαδίκτυο και έχει την δυνατότητα να εκτελεί αρκετές εφαρμογές, τις οποίες μπορεί να επιλέξει ο χρήστης.

Οι βασικές δυνατότητες που προσφέρουν τα έξυπνα τηλέφωνα πηγάζουν από τα κλασσικά Personal Digital Assistants (PDA), τα οποία ήταν ηλεκτρονικές συσκευές που προσέφεραν αρχικά τις λειτουργίες ενός προσωπικού ημερολογίου και στην πορεία μπορούσαν να συνδεθούν με ηλεκτρονικούς υπολογιστές ή με το διαδίκτυο για την ανταλλαγή δεδομένων.

Σταδιακά σε αυτά προστέθηκε και ένα πλήθος από λειτουργίες άλλων φορητών συσκευών, όπως φορητά media players, ψηφιακές φωτογραφικές μηχανές και βιντεοκάμερες, συστήματα παγκόσμιας εύρεσης θέσης (GPS) κα. Αυτός ο συνδυασμός έγινε εφικτός από τις επιπλέον δυνατότητες της συσκευής, η οποία μπορούσε να υποστηρίξει όλα τα παραπάνω.

Αν και δεν είναι απαραίτητο, πολλά σύγχρονα smartphones περιλαμβάνουν επίσης οθόνες αφής υψηλής ανάλυσης και web browsers που εμφανίζουν τυποποιημένες

ιστοσελίδες, καθώς και βελτιστοποιημένες ιστοσελίδες για κινητά. Η πρόσβαση σε δεδομένα υψηλής ταχύτητας παρέχεται μέσω Wi-Fi και μέσω κινητών ευρυζωνικών υπηρεσιών. Τα τελευταία χρόνια, η ταχεία ανάπτυξη στην αγορά των εφαρμογών για κινητά και στο εμπόριο κινητών τηλεφώνων έχει γίνει οδηγός για την ευρεία υιοθέτηση των smartphones.

2.2.2 Λειτουργικό Σύστημα

Το κυριότερο χαρακτηριστικό που ξεχωρίζει τα smartphones είναι το λειτουργικό σύστημα που χρησιμοποιούν. Αυτό αρχικά δημιουργεί έκπληξη στους χρήστες καθώς η έννοια του λειτουργικού είναι συνυφασμένη με τους κλασσικούς υπολογιστές και όχι με φορητές συσκευές. Παρόλα αυτά πρόκειται για ένα πολύ σημαντικό χαρακτηριστικό που μπορεί να αλλάξει πολύ τις λειτουργίες που προσφέρει η συσκευή.

Παλιότερα κάθε εταιρεία που έφτιαχνε κινητά τηλέφωνα έγραφε το δικό της software για να λειτουργούν, το οποίο και κρατούσε κλειστό. Πλέον είναι πολύ συνηθισμένο, ειδικά στο χώρο του Android το λειτουργικό να μην έχει γραφτεί από τον κατασκευαστή.

Ο κυριότερος λόγος για τον οποίο το λειτουργικό είναι σημαντικό είναι το ότι καθορίζει με ποιες εφαρμογές θα είναι συμβατή η συσκευή. Μία συσκευή που έχει εξαιρετικό hardware αλλά δεν έχει διαδεδομένο λειτουργικό δεν είναι τόσο χρήσιμη, γιατί δεν θα υπάρχουν πολλές εφαρμογές για το τηλέφωνο.

Τα βασικά χαρακτηριστικά ενός Mobile OS, δηλαδή ενός λειτουργικού για κινητά είναι πλέον παρόμοια με αυτά ενός κανονικού λειτουργικού συστήματος. Επιπλέον τα πιο διαδεδομένα λειτουργικά κινητών έχουν τις ρίζες τους στα λειτουργικά συστήματα που χρησιμοποιούνται στους υπολογιστές, και συγκεκριμένα στο Linux και στα Windows.

Όμως, αν και έχουν κοινή καταγωγή δεν μπορούν να είναι απόλυτα όμοια. Ο σημαντικότερος στόχος ενός λειτουργικού για κινητά είναι το να καταναλώνει όσον το δυνατόν λιγότερους πόρους, καθώς αυτές οι συσκευές δεν είναι όσο δυνατές είναι οι κανονικοί υπολογιστές και έχουν περιορισμένη διάρκεια μπαταρίας.

2.3 Το λειτουργικό σύστημα Android

2.3.1 Γενικά

Το λειτουργικός σύστημα Android αναπτύχθηκε με πρωτοβουλία της εταιρείας Google σε συνδυασμό με τον οργανισμό OHA (Open Handset Alliance). Είναι ένα λειτουργικό σύστημα βασισμένο στο Linux, με την ίδια βασική αρχιτεκτονική και δυνατότητες.

Το βασικό χαρακτηριστικό του είναι ότι, αν και αναπτύσσεται από μία ομάδα των μεγαλύτερων κατασκευαστών παγκοσμίως, προσφέρεται απολύτως δωρεάν. Αυτός ο συνδυασμός ποιότητας και κόστους έχει συμβάλει στην μεγάλη εξάπλωσή του και στη χρήση του από το μεγαλύτερο μέρος κατασκευαστών.

2.3.2 Εφαρμογές στο Android

Στην πλατφόρμα Android δεν υπάρχει καμία διάκριση μεταξύ ανεξάρτητων και εσωτερικών εφαρμογών όπως η ανταλλαγή μηνυμάτων, τα προγράμματα περιήγησης στο web όπου ο κατασκευαστής παραχωρούσε στον προγραμματιστή δικαίωμα εξουσιοδοτημένης πρόσβασης και γνώσης του εσωτερικού λογισμικού και του firmware της

συσκευής. Οι ίδιες βιβλιοθήκες χρησιμοποιούνται για όλες τις εφαρμογές Android, οι οποίες διαθέτουν πρωτοφανή δικαιώματα πρόσβασης στο υποκείμενο υλικό και μπορούν να διευρυνθούν ή να αντικατασταθούν. Προσαρμόζοντας σε συγκεκριμένους διακομιστές ηλεκτρονικού ταχυδρομείου, παραδείγματος χάριν το Microsoft Exchange ή το Lotus Notes, οι προγραμματιστές Android έχουν πλέον τη δυνατότητα να σχεδιάζουν προγράμματα-πελάτη ηλεκτρονικού ταχυδρομείου δωρεάν.

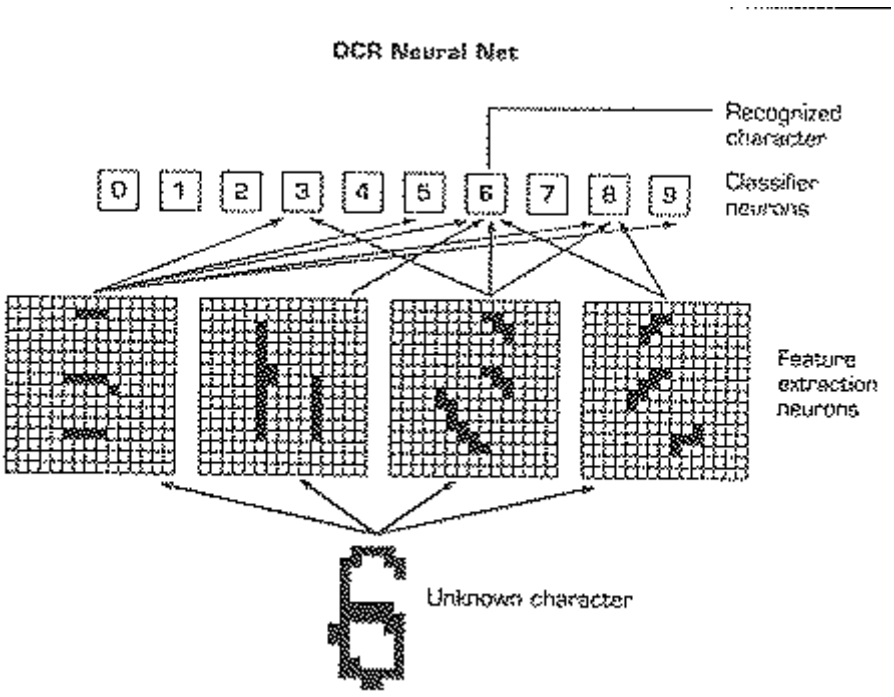
2.3.2.1 Διανομή εφαρμογών

Οι προγραμματιστές πλέον μπορούν να γράψουν, να δημοσιεύσουν, να πουλήσουν οποιοδήποτε είδος εφαρμογής και με οποιοδήποτε μοντέλο εισοδήματος αυτοί επιθυμούν. Μπορούν ακόμη με τις εφαρμογές τους να απευθυνθούν και σε μικρότερα κοινά και όχι μόνο στα μεγαλύτερα που συνήθως οι εταιρείες υπηρεσιών κινητής τηλεφωνίας επιθυμούν. Έτσι υπάρχει, για συγκεκριμένους χρήστες η δυνατότητα κάθετης αγοράς. Υπάρχει πλέον λύση σε διάφορα προβλήματα που αντιμετώπιζαν παλιά οι προγραμματιστές. Παραδείγματος χάριν, λύση στην τακτική των εταιριών να παρέχουν εφαρμογές μόνο για τους πολλούς, λύση στους περιορισμούς των μοντέλων εισοδήματος, τιμολόγησης και της πληρωμής πνευματικών δικαιωμάτων, λύση στον περιορισμό του αριθμού των ανταγωνιστικών εφαρμογών ενός συγκεκριμένου τύπου.

2.4 Οπτική αναγνώριση χαρακτήρων

Ο κλάδος της επιστήμης των υπολογιστών που ασχολείται με την μετατροπή μίας εικόνας σε κείμενο είναι η ανάλυση εικόνας και η οπτική αναγνώριση χαρακτήρων. Για την υπολογιστική αναγνώριση να επεξεργαστεί την εικόνα με έναν τέτοιο αλγόριθμο ώστε να μπορέσει να διαχωρίσει τα γράμματα και να αναγνωρίσει το κάθε ένα από αυτά.

Η εξέλιξη των προγραμμάτων OCR συνεχίζεται σταθερά από το 1974. Αν και η λειτουργία που επιτελούν θεωρείται απλή και αυτονόητη από τους ανθρώπους, είναι αρκετά πολύπλοκη για τους υπολογιστές. Αυτοί οι αλγόριθμοι έχουν πολύ μεγάλο υπολογιστικό κόστος και απαιτούν αρκετό χρόνο για την εκτέλεσή τους.



Εικόνα 2-1 Ανάλυση εικόνας από ένα πρόγραμμα OCR.

Επιπλέον, έχουν αρκετά μειονεκτήματα. Η εξέλιξη τους ώστε να μπορούν να αναγνωρίσουν κείμενο γραμμένο σε διαφορετικές γραμματοσειρές είναι μία σχετικά πρόσφατη εξέλιξη, ενώ παλιότερα χρειαζόταν ειδική εκπαίδευση για την αναγνώριση κάθε γράμματος σε μία συγκεκριμένη γραμματοσειρά. Η αναγνώριση χειρόγραφου κειμένου απέχει ακόμη πάρα πολύ από το να είναι εφικτή.

Χαρακτηριστικό της ικανότητας κάθε αλγορίθμου είναι το ποσοστό επιτυχίας του, το οποίο χαρακτηρίζει το πόσα γράμματα μπορεί να αναγνωρίσει σωστά ο αλγόριθμος επί του συνόλου που του δίνεται. Οι πιο προηγμένοι σύγχρονοι αλγόριθμοι έχουν ποσοστά επιτυχίας πάνω από 99%. Όμως, αν και αυτό το νούμερο ακούγεται υψηλό, έχει πολύ μικρή πρακτική χρησιμότητα.

Για παράδειγμα, αν ένα τέτοιο πρόγραμμα χρησιμοποιηθεί για την μετατροπή ενός βιβλίου σε ηλεκτρονικό κείμενο, η ύπαρξη ενός λάθους σε κάθε εκατό γράμματα κάνει την ανάγνωση εξαιρετικά δυσάρεστη. Ακόμη χειρότερα, υπάρχουν κείμενα που περιέχουν στοιχεία που δεν μπορούν να περιέχουν λάθη, όπως αριθμοί τηλεφώνου ή ταυτότητας. Τέτοια κείμενα είναι αδύνατον να διαβαστούν αυτόματα, καθώς ακόμη και το παραμικρό λάθος καθιστά τις πληροφορίες άχρηστες.

Συνήθως τέτοια προγράμματα συνδυάζονται με εφαρμογές ορθογραφικού ελέγχου. Όμως αυτό δεν είναι ιδανικό, καθώς σε πολλές περιπτώσεις ένα λάθος μπορεί να έχει γίνει σε μία διφορούμενη λέξη ή σε ένα όνομα. Επιπλέον μπορεί να υπάρχει και ένα ορθογραφικό λάθος στο αρχικό κείμενο, του οποίου η διόρθωση να είναι προβληματική ή να αλλάξει το νόημα.

2.4.1 Πηγές

Σχεδόν πάντα οι εικόνες στις οποίες θα εφαρμοστεί μία εφαρμογή OCR δεν βρίσκονται έτοιμες σε ηλεκτρονική μορφή αλλά πρέπει να δημιουργηθούν.

Υπάρχουν δύο βασικοί τρόποι με τους οποίους μπορεί να γίνει αυτό. Ο πρώτος είναι η χρήση ενός σαρωτή και ο δεύτερος η χρήση μίας ψηφιακής φωτογραφικής μηχανής. Σε ιδανικές συνθήκες δεν υπάρχει γενικά κάποιο πλεονέκτημα στην ποιότητα ανάμεσα στις δύο πηγές. Πρακτικά όμως ένας σαρωτής καλής ποιότητας μπορεί να παράγει καλύτερες εικόνες, επειδή ελέγχει πλήρως τις συνθήκες υπό τις οποίες θα γίνει η σάρωση, όπως δεν μπορεί να σαρώσει συγκεκριμένα ήδη χαρτιού (συνήθως δεν μπορεί να χρησιμοποιηθεί πολύ γυαλιστερό ή πλαστικοποιημένο χαρτί).

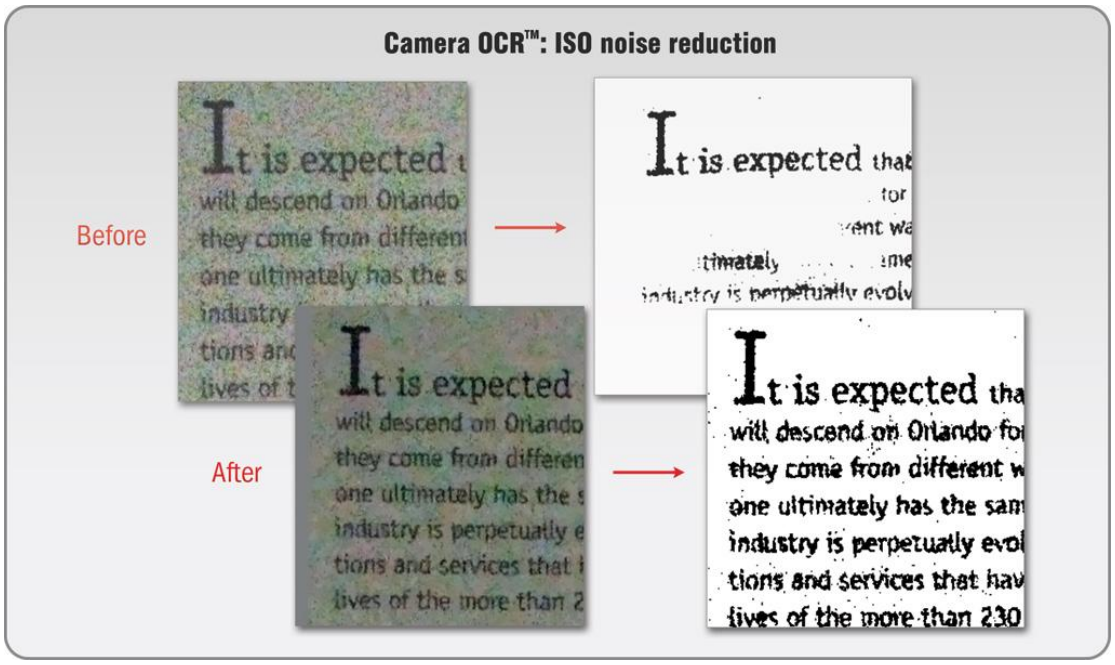
Από την άλλη η διαδικασία σάρωσης είναι πολύ πιο αργή και μπορεί να έχει ατέλειες στις υψηλές αναλύσεις στα φθηνότερα μοντέλα. Οι σαρωτές επίσης δεν είναι αρκετά διαδεδομένες συσκευές και δεν μεταφέρονται εύκολα.

Οι ψηφιακές φωτογραφικές μηχανές έχουν πολύ μεγαλύτερη διάδοση, καθώς αποτελούν τυπικό εξοπλισμό σχεδόν όλων των κινητών τηλεφώνων, και εξαιτίας αυτού μεταφέρονται πολύ εύκολα. Η ποιότητα της εικόνας που παράγεται διαφέρει πάρα πολύ ανάμεσα στις συσκευές.

Αν και μία καλή ψηφιακή μηχανή σε ελεγχόμενο περιβάλλον μπορεί να παράγει αρκετά υψηλής ποιότητας φωτογραφίες, τα αποτελέσματα αλλάζουν πολύ όταν πρόκειται για πραγματικές συνθήκες. Εκεί εισέρχονται πάρα πολύ παράγοντες που παραμορφώνουν το τελικό αποτέλεσμα, όπως η γωνία λήψης, ο φωτισμός και ο ψηφιακός θόρυβος που γίνεται αντιληπτός σε όλες τις ψηφιακές μηχανές (εκτός από τις ακριβότερες).

2.4.2 Επεξεργασία εικόνας

Η εικόνα που επιθυμούμε να αναγνωρίσει μία εφαρμογή OCR έχει σχεδόν πάντα ατέλειες, όμως αναφέραμε στην προηγούμενη παράγραφο. Για αυτό τον λόγο πριν την εκτέλεση της εφαρμογής μία εικόνα πάντα περνάει από επεξεργασία με σκοπό να εξαφανιστούν σε όσο μεγαλύτερο βαθμό γίνεται αυτές οι ατέλειες. Ο λόγος είναι πως πρόκειται για περιττές ή λαθεμένες πληροφορίες που θα μπερδέψουν την εφαρμογή όταν προσπαθεί να αντιστοιχίσει τις εικόνες σε γράμματα.



Εικόνα 2-2 Επεξεργασία εικόνας πριν χρησιμοποιηθεί για αναγνώριση χαρακτήρων.

Όταν λέμε επεξεργασία εικόνας εννοούμε μία σειρά από τεχνικές, με τις ποιο απλές να αναλαμβάνουν απλά να απαλείψουν τα περιττά χρώματα από μία εικόνα ώστε να μείνουν μόνο το άσπρο και το μαύρο. Πέρα από αυτό πρόκειται για τεχνικές που προσπαθούν να απαλείψουν παραμορφώσεις στην αρχική εικόνα, για παράδειγμα όταν η λήψη έχει γίνει υπό γωνία ή αν χρειάζεται να περιστραφεί.

3 Εγκατάσταση Εργαλείων

3.1 Γενικά

Σε αυτό το κεφάλαιο θα δούμε τα βήματα που χρειάζεται να ακολουθήσουμε για να ρυθμίσουμε το περιβάλλον εργασίας για την ανάπτυξη της εφαρμογής μας.

3.2 Απαραίτητες εφαρμογές

Το πρόγραμμα χρειάζεται την εγκατάσταση του περιβάλλοντος ανάπτυξης eclipse μαζί με το κατάλληλο plugin για την ανάπτυξη εφαρμογών Android. Επίσης χρειάζεται την εγκατάσταση της πλατφόρμας Android.

Για το μέρος του server χρειάζεται ο Apache Tomcat server για την εκτέλεση του servlet που χρησιμοποιείται για το διαδικτυακό μέρος της εφαρμογής. Για αυτό προτιμήσαμε το πρόγραμμα ανάπτυξης εφαρμογών Netbeans αντί του eclipse λόγω της πιο απλής διασύνδεσης με τον Tomcat.

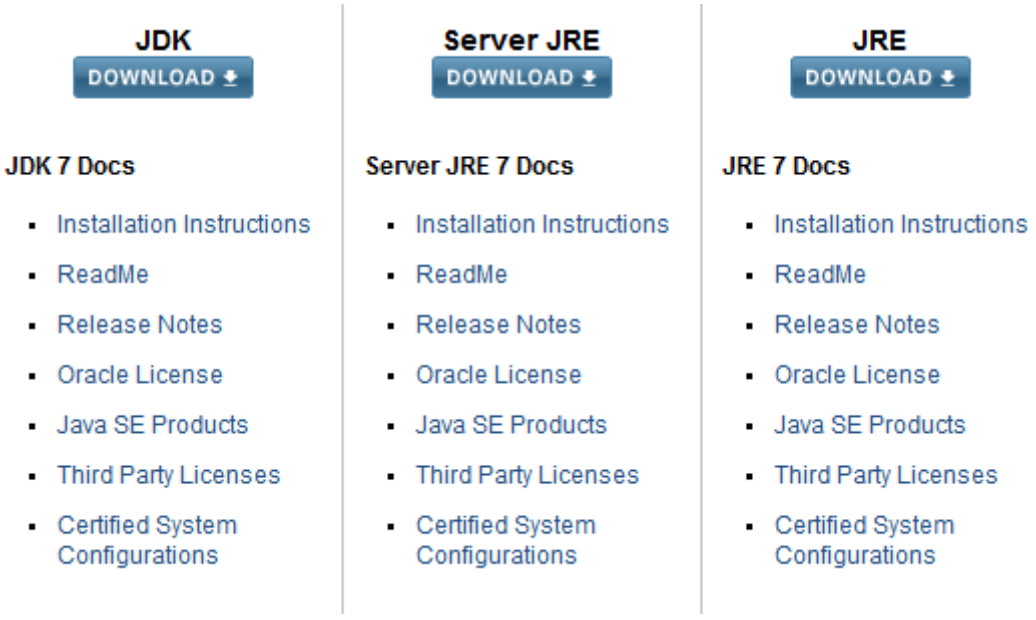
Επιπλέον χρειάζεται η βιβλιοθήκη tesseract για την αναγνώριση των χαρακτήρων και μία βιβλιοθήκη για την διασύνδεση με το Microsoft Translate.

3.3 Εγκατάσταση Java JDK

Όλα τα εργαλεία που χρησιμοποιούμε και η εφαρμογή που δημιουργούμε είναι γραμμένα σε Java. Για να τα χρησιμοποιήσουμε πρέπει να έχουμε εγκαταστήσει το Java JDK (Standard Development Kit). Το Java Standard Development Kit περιέχει τον compiler και τις βιβλιοθήκες της Java, καθώς και την Java Virtual Machine η οποία χρησιμοποιείται για την εκτέλεση των προγραμμάτων.

Κατεβάζουμε το Java JDK από τη διεύθυνση:

http://www.oracle.com/technetwork/java/javase/downloads/index.html



Εικόνα 3-1 Διαφορετικές εκδόσεις εργαλείων της Java

Έπειτα επιλέγουμε την κατάλληλη έκδοση του αρχείου για το λειτουργικό μας σύστημα. Αξίζει να σημειωθεί πως όλα τα εργαλεία είναι διαθέσιμα σε 32 και 64-bit εκδόσεις, αλλά ανάλογα με την έκδοση της Java χρειάζεται να κατεβάσουμε και την ίδια έκδοση για τα υπόλοιπα εργαλεία.

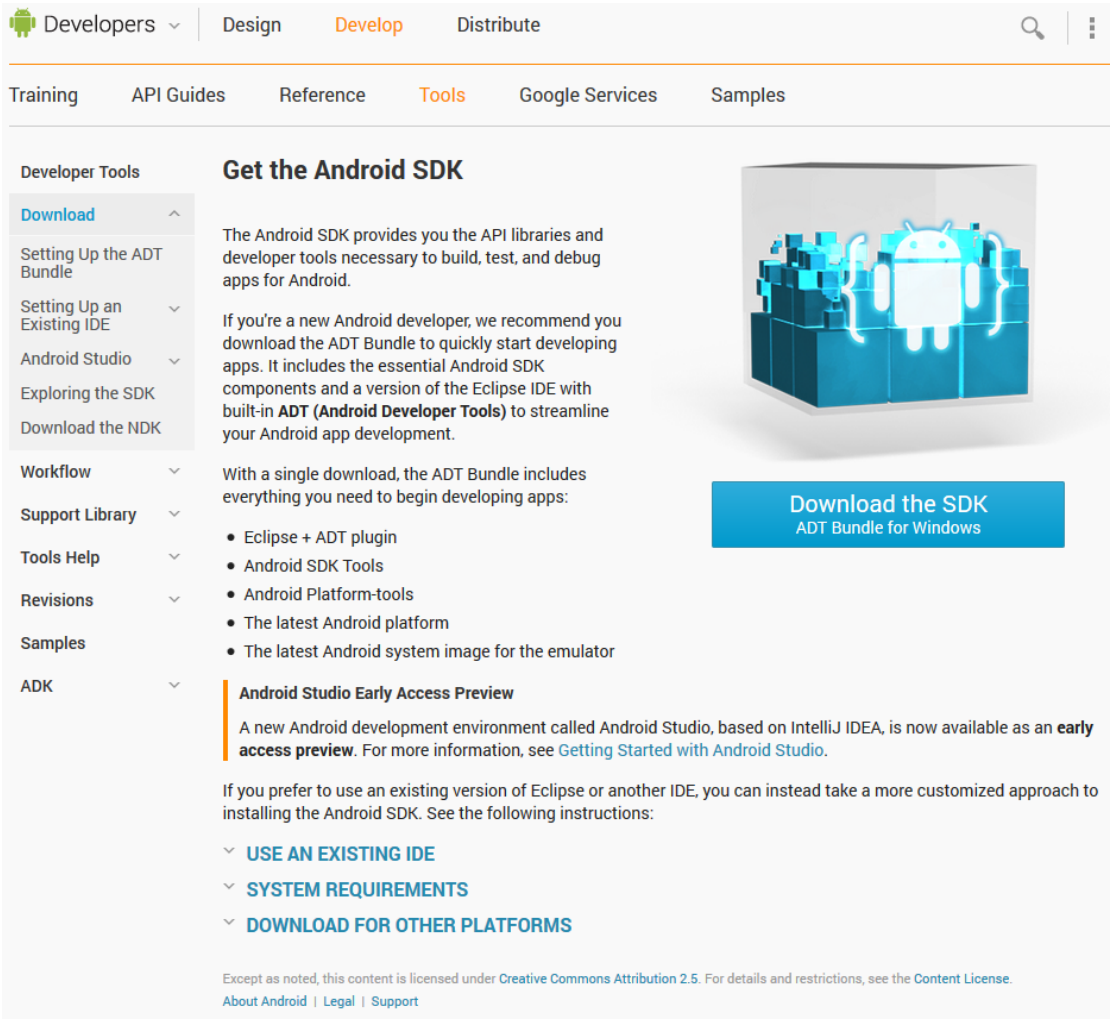
Εμείς επιλέξαμε τις 32-bit εκδόσεις, κυρίως επειδή κατά την περίοδο ανάπτυξης υπήρχαν διάφορα θέματα συμβατότητας με τις 64-bit εκδόσεις.

3.4 Εγκατάσταση Android ADT

Η εταιρεία Google παρέχει στους προγραμματιστές τα απαραίτητα εργαλεία για την ανάπτυξη εφαρμογών σε Android. Αυτά είναι διαθέσιμα από τη διεύθυνση

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

Εκεί φαίνεται η αρχική σελίδα του Android ADT (Android Development Tools), όπου βρίσκονται συγκεντρωμένα τα σχετικά εργαλεία.



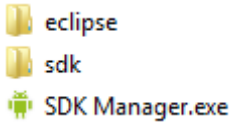
Εικόνα 3-2 Η σελίδα με το Android ADT

Πατώντας τον σύνδεσμο Download the ADT παίρνουμε ένα συμπιεσμένο αρχείο zip, το οποίο αποσυμπιέζουμε σε όποιον φάκελο θέλουμε και δεν χρειάζεται εγκατάσταση. Αυτό περιέχει μία εγκατάσταση του eclipse μαζί με το κατάλληλο plugin για Android και τα εργαλεία και τις βιβλιοθήκες για ανάπτυξη.

Παλιότερα η Google πρόσφερε ξεχωριστά το plugin και έπρεπε να εγκατασταθεί σε μία υπάρχουσα εγκατάσταση του eclipse. Πλέον δίνει και τα δύο σε ένα έτοιμο πακέτο ώστε να αποφευχθεί η χρονοβόρα διαδικασία εγκατάστασης.

Οι βιβλιοθήκες για ανάπτυξη ονομάζονται συνολικά Android platform. Αυτές περιέχουν όλες τις έτοιμες βιβλιοθήκες για τη διασύνδεση μίας εφαρμογής με το λειτουργικό.

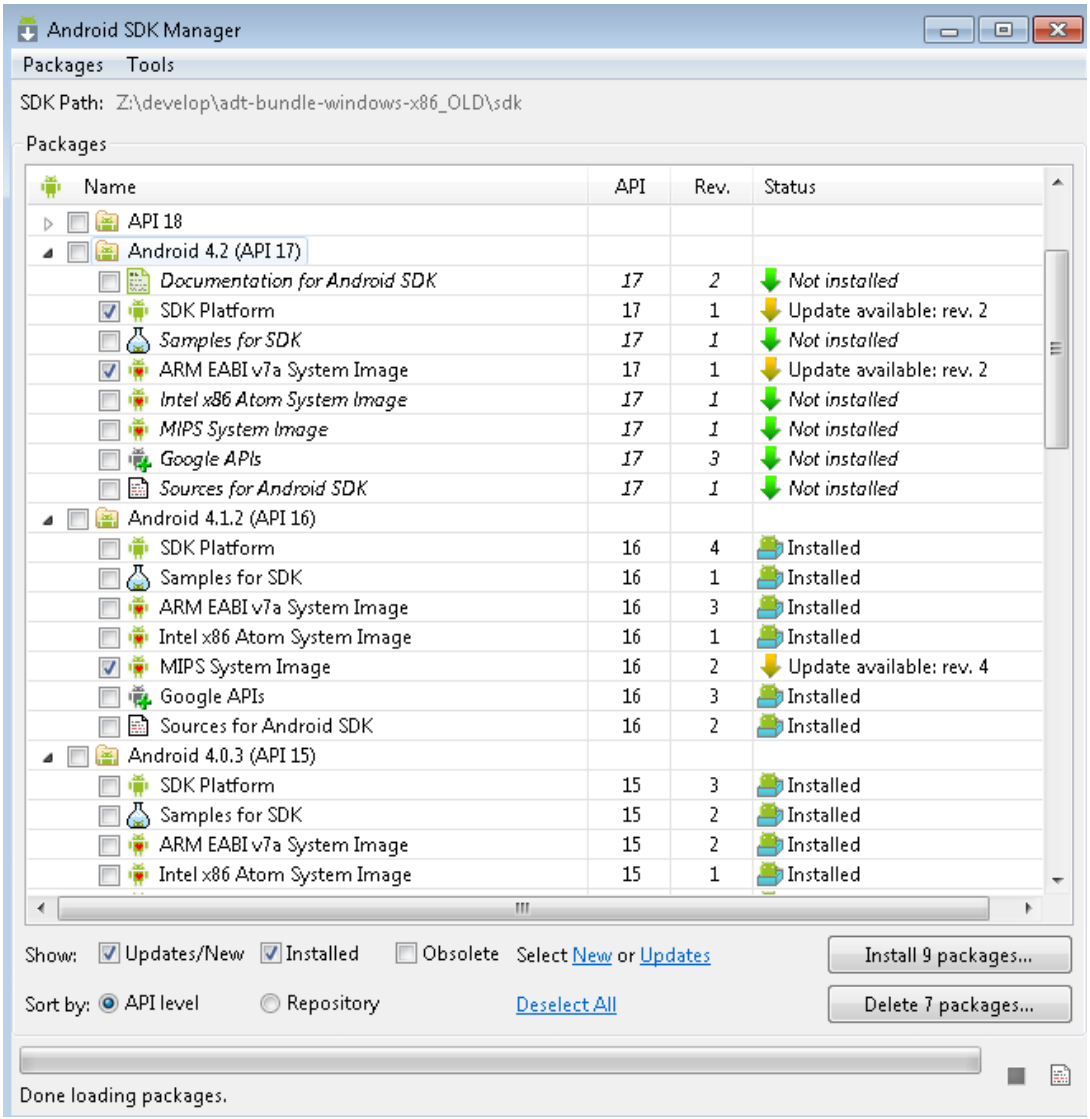
Οι φάκελοι φαίνονται παρακάτω:



Εικόνα 3-3 Οι φάκελοι εγκατάστασης του Andoid SDK

Ο φάκελος eclipse περιέχει το eclipse, ο φάκελος sdk περιέχει την πλατφόρμα και το SDK manager είναι ένα εργαλείο διαχείρισης για τις εγκαταστημένες βιβλιοθήκες.

Για κάθε έκδοση του Android χρειάζεται και μία διαφορετική έκδοση της πλατφόρμας. Αρχικά είναι εγκαταστημένη μόνο η τελευταία έκδοση, το API 16 που είναι για την έκδοση 4.1.2, το οποίο ήταν το πιο πρόσφατο όταν είχε γίνει η εγκατάσταση της εφαρμογής.



Εικόνα 3-4 Τα εγκαταστημένα πακέτα

Κάθε έκδοση είναι θεωρητικά συμβατή με τις παλιότερες, πχ αν μία εφαρμογή έχει γραφτεί χρησιμοποιώντας το API 16 (Android 4.1.2) τότε μπορεί να τρέχει σε συσκευές με παλιότερες εκδόσεις, εκτός και αν η εφαρμογή προσδιορίζει με διαφορετικό τρόπο τη συμβατότητά της, αν χρησιμοποιεί δυνατότητες που δεν μπορούν να εξομοιωθούν σε παλιότερες εκδόσεις ή αν η έκδοση του API έχει πάψει να υποστηρίζεται. Επιπλέον μία εφαρμογή που είναι γραμμένη σε μία παλιότερη έκδοση μπορεί να τρέξει θεωρητικά και σε νεότερες.

Για την ανάπτυξη της εφαρμογής χρησιμοποιήσαμε την (τρέχουσα) έκδοση 19 του API, και ορίσαμε σαν ελάχιστη πλατφόρμα την έκδοση 10 που αντιστοιχεί στο Android 2.3.3. Η επιλογή έγινε μελετώντας τους χρήστες κάθε εγκατάστασης, όπως δίνονται από στατιστικά της εταιρεία Google.

Version	Codename	API	Distribution
2.2	Froyo	8	1.0%
2.3.3 - 2.3.7	Gingerbread	10	16.2%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	13.4%
4.1.x	Jelly Bean	16	33.5%
4.2.x		17	18.8%
4.3		18	8.5%
4.4	KitKat	19	8.5%

Εικόνα 3-5 Τα μερίδια αγοράς διαφόρων εκδόσεων του Android.

Για να μπορούμε να δουλέψουμε με μία παλιότερη έκδοση του API είναι απαραίτητο να την έχουμε εγκαταστήσει, αλλιώς όταν ανοίγουμε το project από το eclipse εμφανίζεται το παρακάτω μήνυμα:

```
Android
[2013-12-29 15:00:23 - CameraTranslator] Unable to resolve target 'android-17'
[2013-12-29 15:00:23 - CameraTranslator] Unable to resolve target 'android-17'
```

Εικόνα 3-6 Η έκδοση API του project δεν υπάρχει εγκαταστημένη

Για την εγκατάσταση μίας διαφορετικής έκδοσης του API χρειάζεται στον SDK manager να επιλέξουμε τα κατάλληλα πακέτα προς εγκατάσταση. Δεν χρειάζονται όλες οι επιλογές. Η μοναδικές εκδόσεις που χρειαζόμαστε είναι αυτή στην οποία έχουμε γράψει το πρόγραμμα. Παλιότερες εκδόσεις χρησιμεύουν στο να μπορούμε να εξομοιώσουμε συσκευές με τέτοιο λογισμικό, για έλεγχο συμβατότητας κατά την ανάπτυξη.

☐ Android 4.2.2 (API 17)			
☑ SDK Platform	17	2	Not installed
☐ Samples for SDK	17	1	Not installed
☑ ARM EABI v7a System Image	17	2	Not installed
☐ Intel x86 Atom System Image	17	1	Not installed
☐ MIPS System Image	17	1	Not installed
☐ Google APIs	17	3	Not installed
☐ Sources for Android SDK	17	1	Not installed

Εικόνα 3-7 Τα κατάλληλα πακέτα εγκατάστασης για το API 17

Η εγκατάσταση ξεκινάει πατώντας το Install Selected Packages. Παράλληλα με τις επιλογές μας ανανεώνονται αυτόματα και οι πιο πρόσφατες εκδόσεις των υπαρχόντων πακέτων.

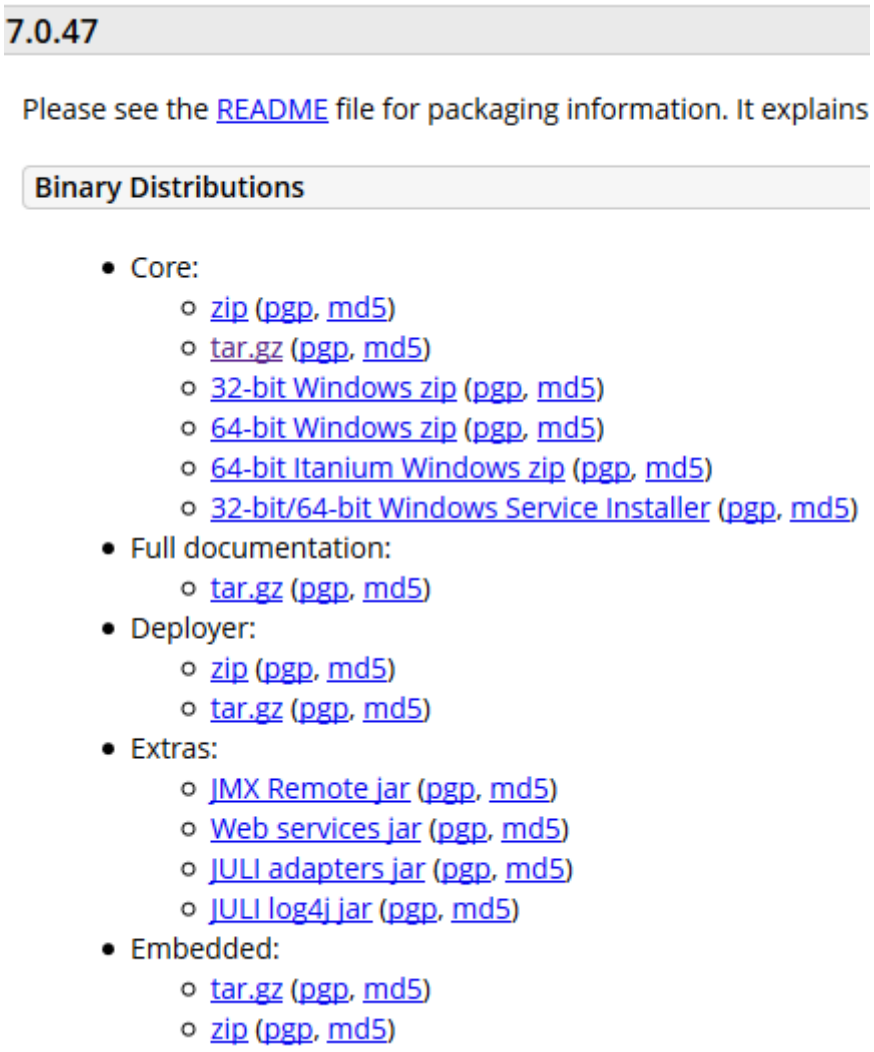
Μετά από αυτό αρκεί να εκτελέσουμε το eclipse (από το eclipse.exe που βρίσκεται μέσα στο φάκελο eclipse) ώστε να ξεκινήσει το πρόγραμμα ανάπτυξης.

3.5 Εγκατάσταση Tomcat

Η εγκατάσταση του Tomcat είναι αρκετά απλή και γίνεται από την διεύθυνση

http://tomcat.apache.org/

Εκεί επιλέγουμε την έκδοση που θέλουμε. Εμείς χρησιμοποιήσαμε την έκδοση 7 που είναι η τελευταία σταθερή έκδοση. Θεωρητικά δεν θα υπάρχουν θέματα συμβατότητας ανάμεσα στις διαφορετικές εκδόσεις.



Εικόνα 3-8 Οι επιλογές για το κατέβασμα του Tomcat

Αφού επιλέξουμε την έκδοση μπορούμε να πάρουμε το πρόγραμμα με διαφορετικές μορφές. Εμείς θέλουμε την επιλογή tar.gz, στην οποία περιέχονται όλα τα αρχεία του Tomcat.

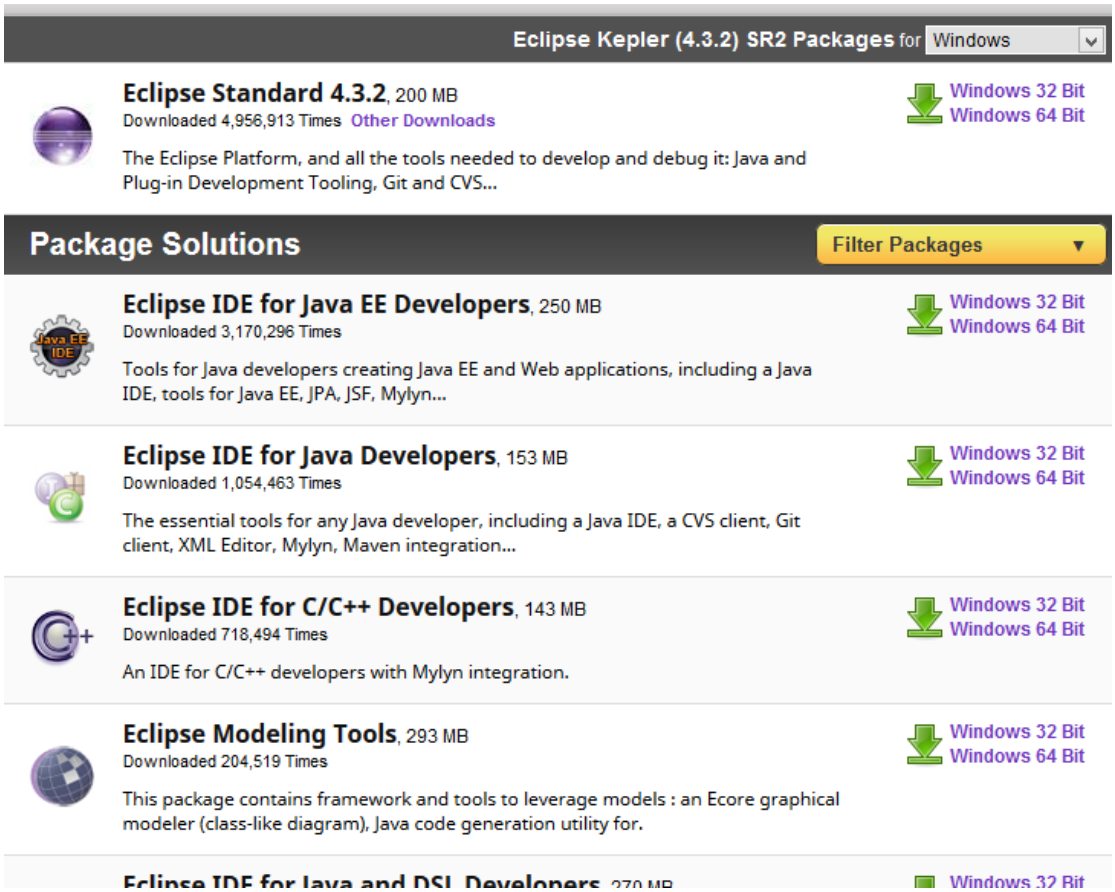
Οι επιπλέον επιλογές δίνουν τον server με διάφορα επιπλέον προγράμματα που καθορίζουν τον τρόπο εκτέλεσης αλλά δεν χρειαζόμαστε τίποτα επιπλέον γιατί η διαχείριση του server θα γίνεται μέσα από τα εργαλεία ανάπτυξής που θα χρησιμοποιήσουμε.

3.6 Εγκατάσταση eclipse

Για το τελικό web app που γράψαμε χρησιμοποιήσαμε το eclipse. Αν και το eclipse περιέχεται μέσα στο Android ADT που περιγράψαμε παραπάνω, εκείνη η έκδοση έχει τις ρυθμίσεις που χρειάζονται για την ανάπτυξη προγραμμάτων Android. Για να μπορεί να υποστηρίξει την ανάπτυξη επιπλέον τύπων project, όπως τα web project που χρειαζόμαστε χρειάζονται επιπλέον plugins και ρυθμίσεις.

Καθώς τα παραπάνω είναι πολύπλοκο να γίνουν, και από φόβο ότι μπορεί να υπάρξουν συγκρούσεις με τις υπάρχουσες ρυθμίσεις, αποφασίσαμε να χρησιμοποιήσουμε ένα διαφορετικό αντίγραφο του eclipse για την εφαρμογή του Tomcat. Αυτό μπορούμε να το κατεβάσουμε από τη διεύθυνση:

https://www.eclipse.org/downloads/



Χρησιμοποιήσαμε το Eclipse IDE for Java EE Developers, που είναι μία έκδοση με τα κατάλληλα plugins και ρυθμίσεις για την ανάπτυξη διαδικτυακών εφαρμογών. Όπως και παραπάνω επιλέξαμε την έκδοση 32 Bit.

Με αυτό κατεβάζουμε ένα συμπιεσμένο αρχείο, το οποίο αποσυμπιέζουμε στον φάκελο της επιλογής μας.

Όταν ξεκινάει το eclipse ζητάει να ορίσουμε έναν φάκελο για να τον χρησιμοποιήσει για workspace, στο οποίο θα αποθηκεύει τα project και τις ρυθμίσεις του. Ο φάκελος που θα ορίσουμε πρέπει να είναι διαφορετικός από τον φάκελο που

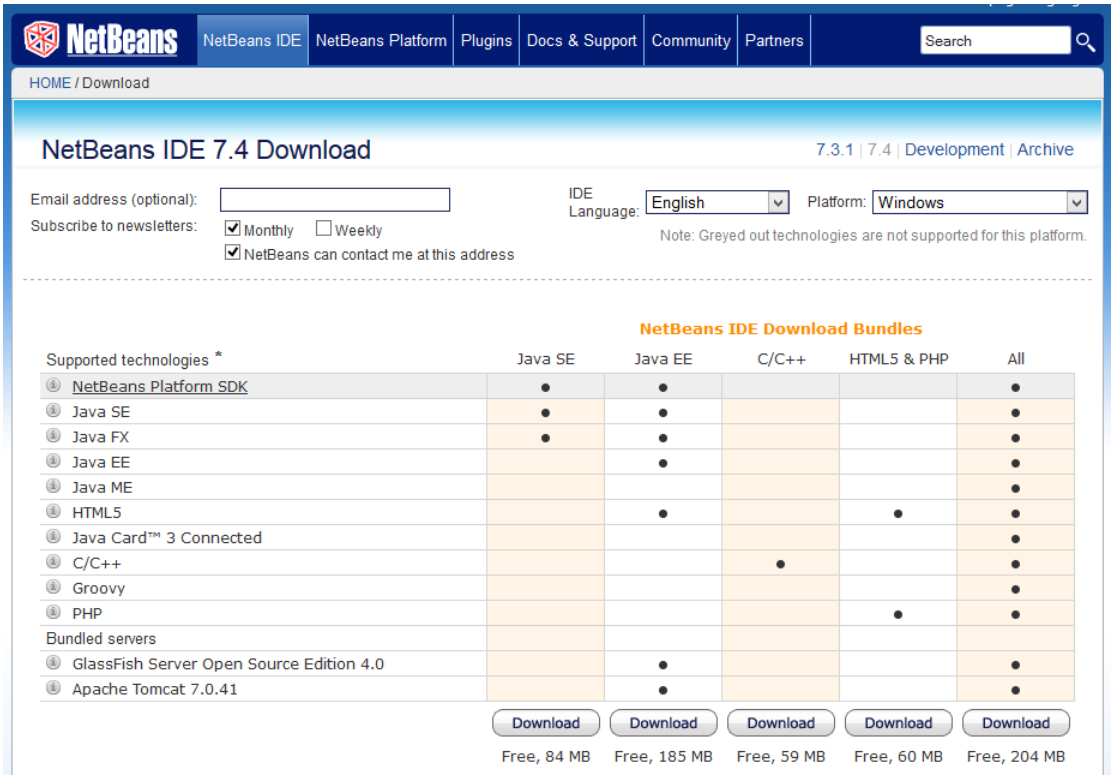
χρησιμοποιήσαμε από τον φάκελο για το workspace που ορίσαμε στο eclipse του ADT, αλλιώς θα υπάρχουν συγκρούσεις μεταξύ τους.

3.7 Εγκατάσταση NetBeans

Για την ανάπτυξη των servlets χρησιμοποιήσαμε αρχικά το περιβάλλον NetBeans και έπειτα μεταφέραμε το project στο eclipse. Μπορούμε να κατεβάσουμε το NetBeans από τη διεύθυνση:

https://netbeans.org/downloads/

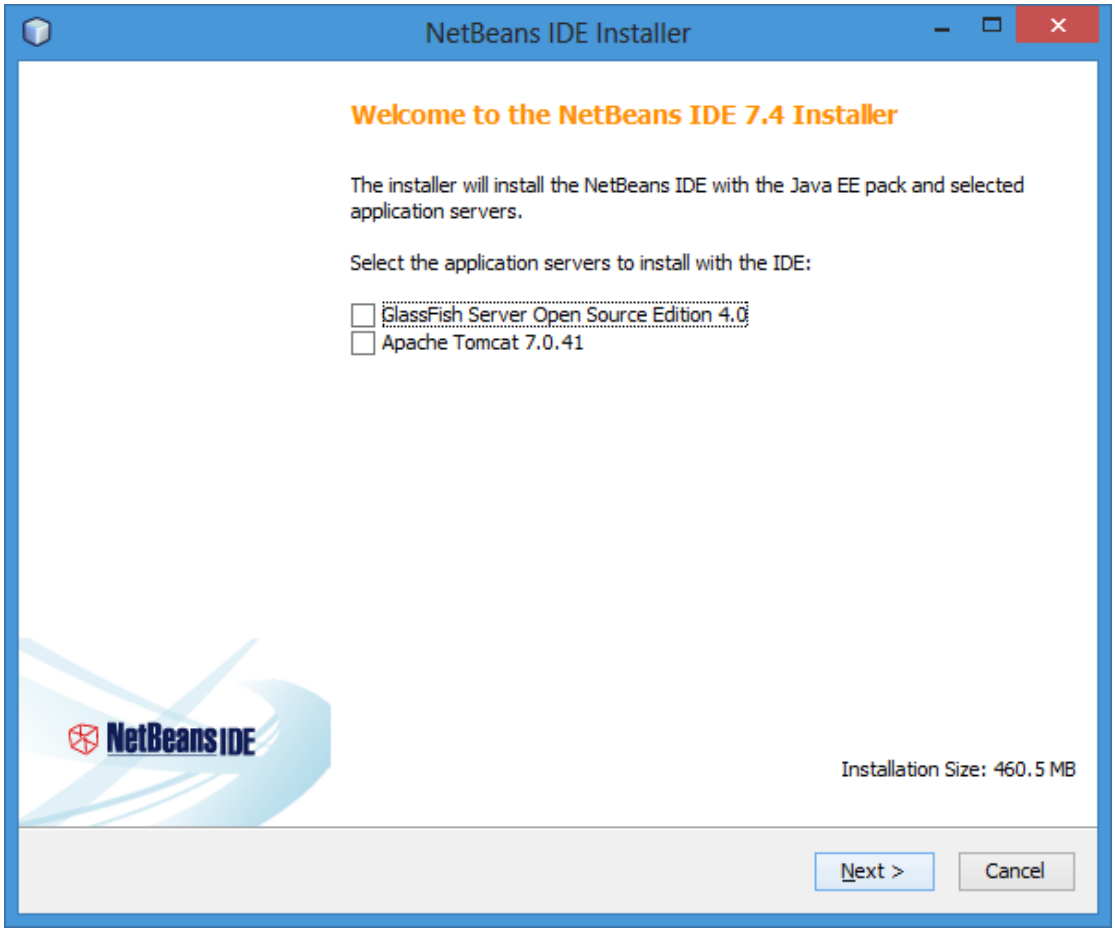
Όπως και ο Tomcat έχει και αυτό διάφορες επιλογές.



Εικόνα 3-9 Τα διαφορετικά πακέτα του NetBeans

Θέλουμε το πακέτο Java EE που επιτρέπει την ανάπτυξη εφαρμογών servlets.

Αφού τρέξουμε το εκτελέσιμο ξεκινάει η διαδικασία εγκατάστασης, η οποία στην αρχή εμφανίζει την παρακάτω οθόνη:



Εικόνα 3-10 Επιλογή servers που θα εγκαταστήσει το NetBeans

Το πακέτο που κατεβάσαμε περιέχει δύο servers, τον Tomcat και τον Glassfish. Δεν χρειαζόμαστε κανένα από τους δύο, οπότε βγάζουμε αυτές τις επιλογές. Ο Glassfish είναι περιττός για τη δουλειά που τον θέλουμε, ενώ έχουμε ήδη τον Tomcat.

Ο λόγος που κάναμε ανεξάρτητα την εγκατάστασή του είναι πως μετά θα χρειαστεί να τροποποιήσουμε τα περιεχόμενα του φακέλου του ώστε να λειτουργήσει η βιβλιοθήκη Tesseract. Η εγκατάσταση που κάνει ο Tomcat του NetBeans παρουσιάζει διάφορα προβλήματα σε συστήματα Windows εξαιτίας του τρόπου με τον οποίο προστατεύονται τα αρχεία προγραμμάτων.

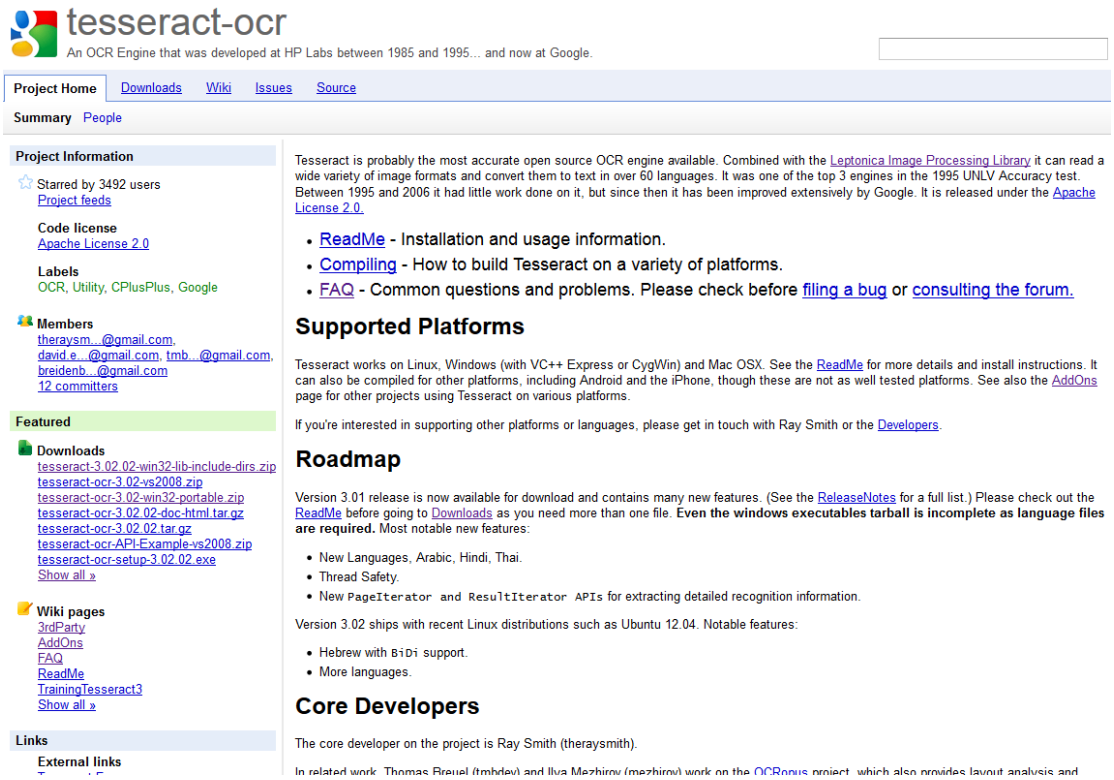
3.8 Εγκατάσταση Tesseract

Η βιβλιοθήκη Tesseract είναι μία βιβλιοθήκη ανοικτού κώδικα για οπτική αναγνώριση χαρακτήρων (OCR – Optical Character Recognision). Η ανάπτυξή της ξεκίνησε το 1985 στα εργαστήρια της Hewlett Packard και από το 2006 υποστηρίζεται από το Google.

Η βιβλιοθήκη είναι γραμμένη στη γλώσσα C++. Η συνεργασία της με προγράμματα Java είναι δυνατή μέσω της δυνατότητας JNI (Java Native Interface) που επιτρέπει την εκτέλεση προγραμμάτων που έχουν γίνει compile για ένα λειτουργικό σύστημα. Αν και η διαδικασία είναι πολύπλοκη, μαζί με το Tesseract δίνεται και η βιβλιοθήκη Java Tess4J, με τη χρήση της οποίας μπορούμε να χειριστούμε τη Tesseract σαν να είναι ένα αντικείμενο Java, χωρίς επιπλέον ρυθμίσεις.

Η αρχική σελίδα του project είναι η:

http://code.google.com/p/tesseract-ocr/



Εικόνα 3-11 Η αρχική σελίδα του Tesseract

Από εκεί κατεβάσαμε δύο αρχεία, τα tesseract-3.02.02-win32-lib-include-dirs.zip και tesseract-3.02.02-win32-lib-include-dirs.zip. Το πρώτο περιέχει τα αρχεία dll που περιέχουν την ίδια τη βιβλιοθήκη, που είναι τα αρχεία C++ που έχουν γίνει compile για Windows. Το δεύτερο περιέχει κάποια αρχεία εκπαίδευσης που περιέχουν κατάλληλα δεδομένα ώστε το Tesseract να μπορεί να αναγνωρίσει χαρακτήρες σε άλλες γλώσσες.

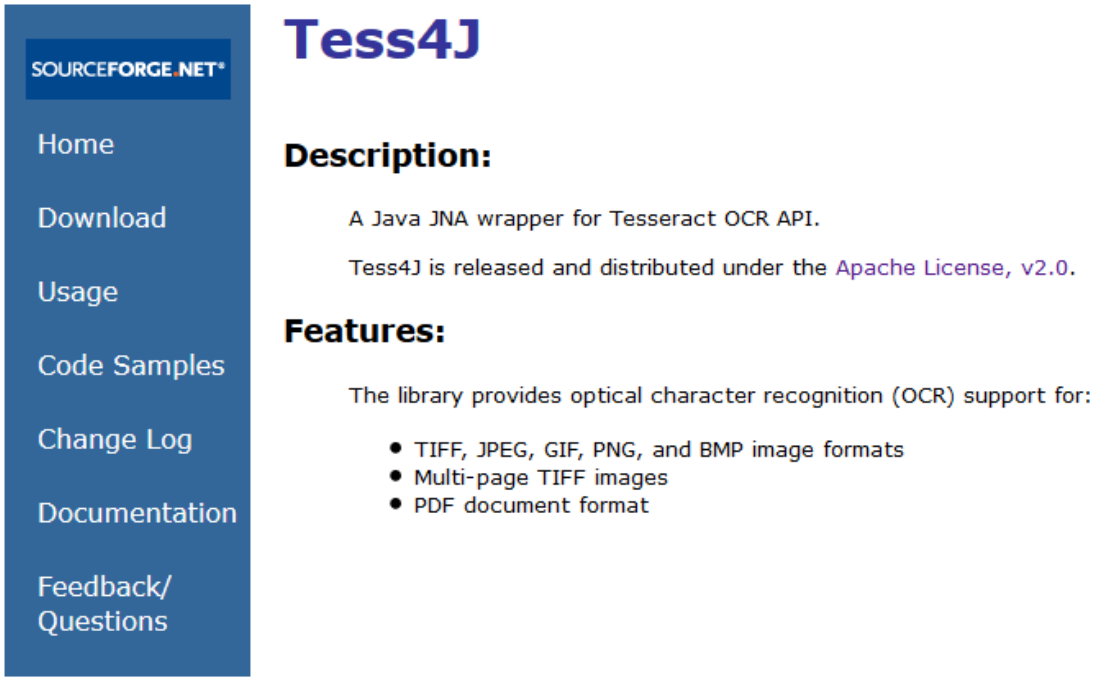
Η διαδικασία συνεργασίας του Tesseract με τον Tomcat που παρουσιάζουμε παρακάτω ήταν από τα πιο δύσκολα και χρονοβόρα τμήματα της πτυχιακής, κυρίως εξαιτίας της έλλειψης συγκεκριμένων οδηγιών. Επίσης ήταν και ο λόγος που παρόλο που χρησιμοποιήσαμε το NetBeans για την ανάπτυξη των servlets εν τέλει το μεταφέραμε στο eclipse για την εκτέλεσή του.

3.9 Εγκατάσταση Tess4J

Το Tess4J είναι μία βιβλιοθήκη που επιτρέπει τη χρήση του Tesseract μέσα από προγράμματα Java σαν να είναι και αυτό μία κλάση Java. Στόχος είναι να κάνει τελείως διαφανή τη χρήση του Tesseract, χωρίς να πρέπει να προσδιορίσουμε πως είναι μία βιβλιοθήκη γραμμένη σε C++.

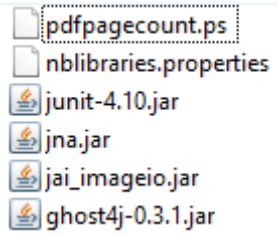
Το Tess4J βρίσκεται στη διεύθυνση:

http://tess4j.sourceforge.net/



Εικόνα 3-12 Η αρχική σελίδα του Tess4J

Από εκεί κατεβάζουμε το Tess4J-1.2-src.zip. Από αυτό χρειαζόμαστε κάποια αρχεία jar που περιέχει, συγκεκριμένα όσα βρίσκονται μέσα στους φακέλους lib και dist.



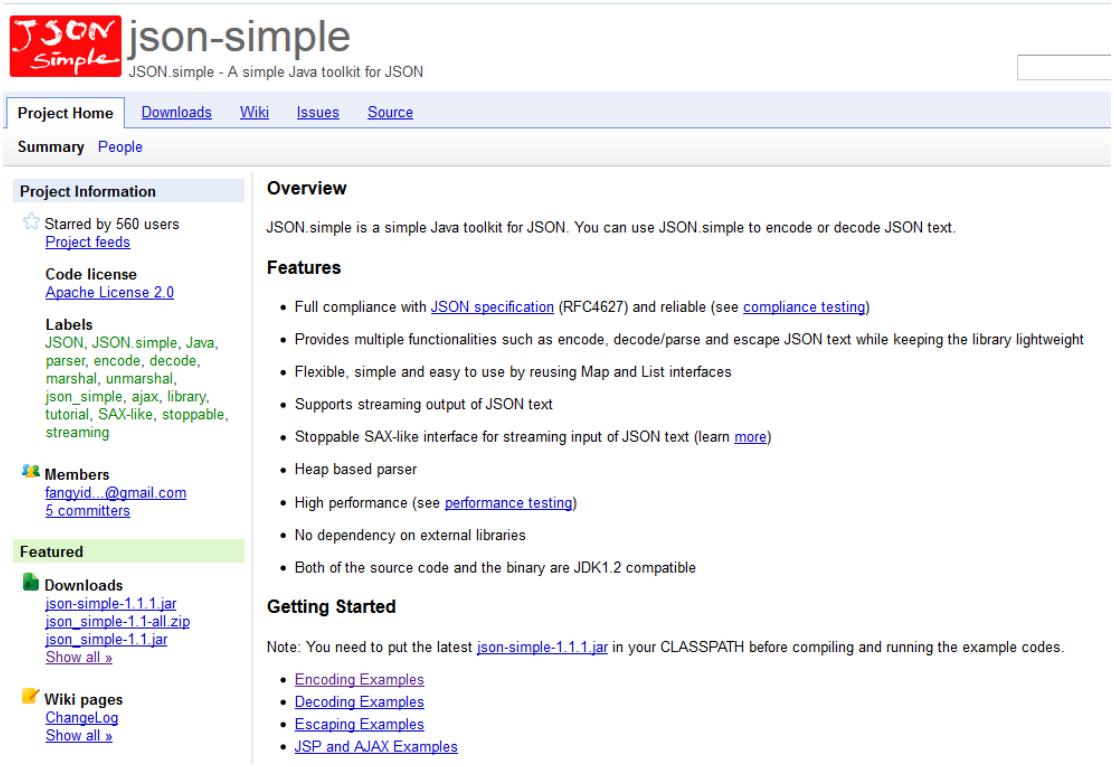
Εικόνα 3-13 Τα περιεχόμενα του φακέλου lib του Tess4J

Αυτά πρέπει να γραφτούν στον κατάλληλο φάκελο του project, όπως θα εξηγήσουμε στην ανάλυση της εφαρμογής.

3.10 Εγκατάσταση json-simple

Οι δύο εφαρμογές επικοινωνούν μεταξύ τους ανταλλάσοντας μηνύματα σε JSON. Για την δημιουργία των απαντήσεων JSON χρησιμοποιούμε τη βιβλιοθήκη JSON Simple, που βρίσκεται στη διεύθυνση:

http://code.google.com/p/json-simple/



Εικόνα 3-14 Η σελίδα της βιβλιοθήκης json-simple

Από εκεί κατεβάζουμε το αρχείο jar json-simple-1.1.1.jar, το οποίο αντιγράφουμε στον κατάλληλο φάκελο του project, όπως θα προσδιορίσουμε στην ανάλυση της εφαρμογής.

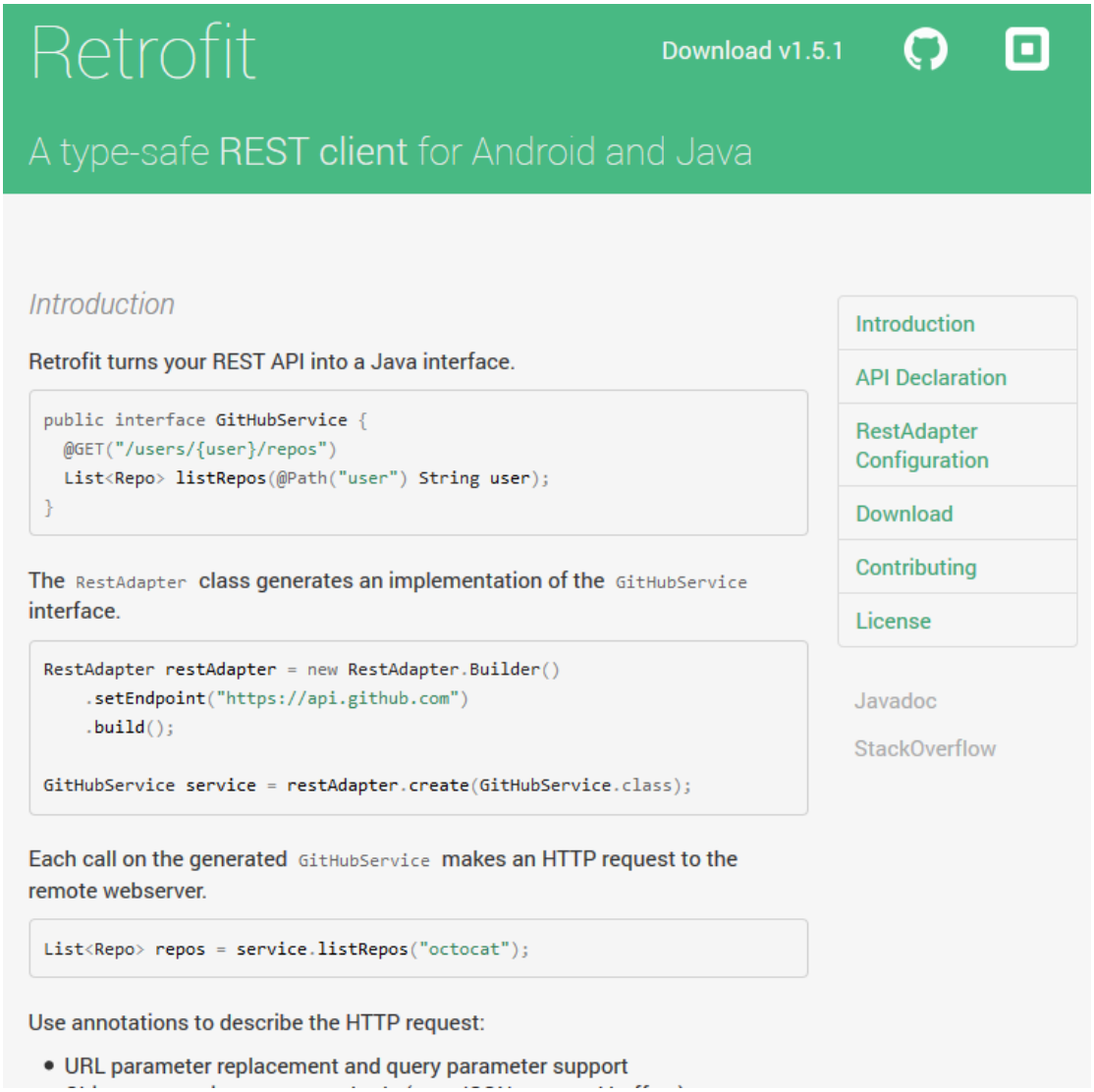
3.11 Εγκατάσταση Retrofit

Η εφαρμογή μας χρειάζεται να μπορεί να επικοινωνήσει με έναν server ώστε να του στείλει την εικόνα και να πάρει την απάντηση. Αν και το Android παρέχει κάποιες βασικές κλάσεις που αναλαμβάνουν αυτή τη διαδικασία, αυτές έχουν διάφορα προβλήματα και η χρήση τους είναι αρκετά δύσκολη.

Για αυτό το λόγο χρησιμοποιούμε τη βιβλιοθήκη Retrofit, η οποία είναι αρκετά πιο εύκολη στη χρήση της και διορθώνει bugs του Android.

Η αρχική σελίδα της βιβλιοθήκης βρίσκεται στο:

http://square.github.io/retrofit/



Εικόνα 3-15 Η αρχική σελίδα του Retrofit.

Από εδώ κατεβάζουμε το κατάλληλο αρχείο jar, το οποίο βάζουμε μέσα στο project μας.

4 Ανάλυση εφαρμογής

4.1 Γενικά

Η εφαρμογή αποτελείται από δύο διαφορετικά κομμάτια, την εφαρμογή Android και την εφαρμογή με τα servlets. Αυτά είναι τελείως ανεξάρτητα μεταξύ τους από την άποψη πως είναι δύο διαφορετικά project που δεν χρειάζεται να γίνουν κάποιες ρυθμίσεις για να μπορεί να δει το ένα το άλλο. Ο λόγος είναι πως επικοινωνούν απλά μέσω δικτύου, που σημαίνει πως η εφαρμογή του Android αρκεί να ξέρει τη διεύθυνση του server που τρέχει το δεύτερο project για να λειτουργήσει.

Επιπλέον χρησιμοποιούμε μια ομάδα από βιβλιοθήκες για να υποστηρίξουμε τις λειτουργίες που απαιτούνται. Παρακάτω θα δούμε αρχικά πώς αυτά τα εργαλεία χρησιμοποιούνται ανεξάρτητα, σε μεμονωμένα project java, και κατόπιν θα τα ενώσουμε σε ένα τελικό web application. Έπειτα θα αναλύσουμε ξεχωριστά τα δύο project της εργασίας, βασισμένοι σε αυτή τη δομή.

4.2 Project βιβλιοθηκών

4.2.1 Επικοινωνία με το Microsoft Translator

Για την μετάφραση των κειμένων χρησιμοποιούμε την υπηρεσία Microsoft Translator. Αν και η πρώτη επιλογή μας ήταν η πιο γνωστή και παλιότερη υπηρεσία Google Translate, η χρήση της μέσω τρίτων εφαρμογών δεν είναι δωρεάν, οπότε καταλήξαμε στην υπηρεσία της Microsoft η οποία εν τέλει ικανοποιεί πλήρως τις ανάγκες μας.

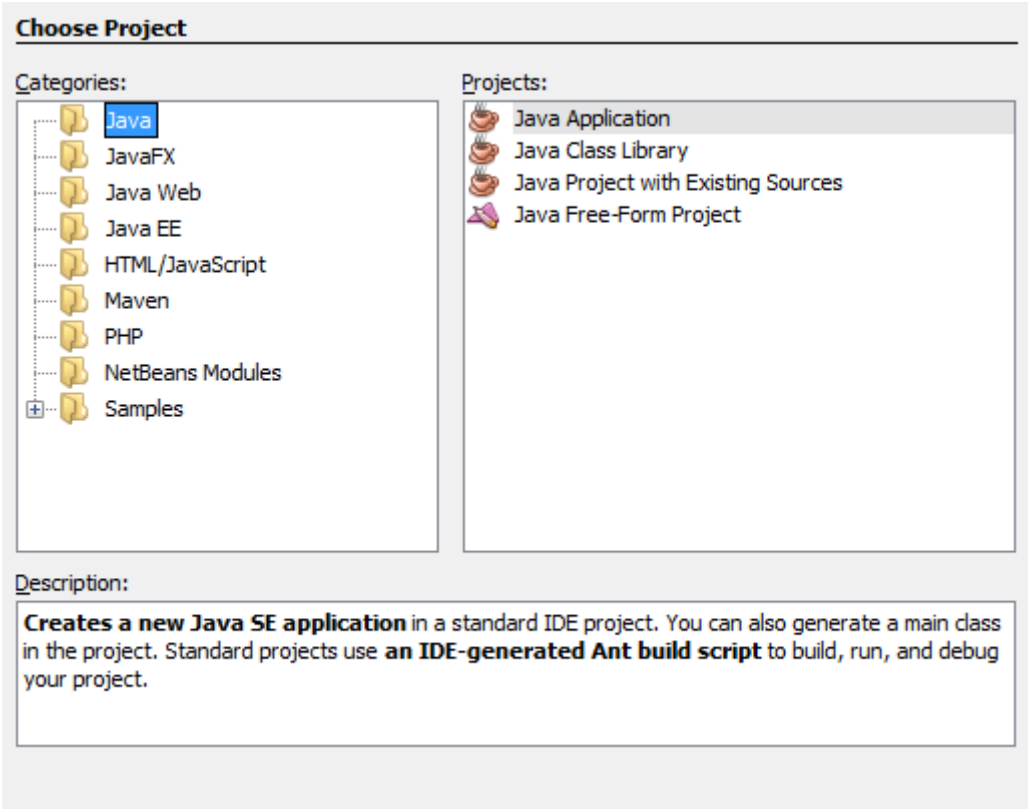
Για να έχουμε δικαίωμα χρήσης της υπηρεσίας είναι απαραίτητη η δωρεάν δημιουργία ενός προσωπικού κλειδιού για την χρήση του API. Για αυτό χρειάζεται η ύπαρξη ή η δωρεάν λειτουργία ενός λογαριασμού email στις υπηρεσίες της Microsoft, όπως το Hotmail ή το live.com ή το outlook.com. Έπειτα μπορούμε να δημιουργήσουμε από τη σελίδα

http://www.bing.com/developers/appids.aspx

Η χρήση της υπηρεσίας γίνεται μέσω ενός RESTful HTTP API. Αυτό σημαίνει πως τα προγράμματα που τη χρησιμοποιούν πρέπει να στείλουν δεδομένα στους servers της υπηρεσίας με μία αίτηση HTTP και να διαβάσουν την απάντηση. Για την αυτοματοποίηση της διαδικασίας χρησιμοποιήσαμε μία έτοιμη βιβλιοθήκη που κρύβει την αποστολή των δεδομένων μέσω δικτύου, την microsoft-translator-java-api.

Εδώ θα δούμε ένα βασικό project που δείχνει τη χρήση αυτής της βιβλιοθήκης.

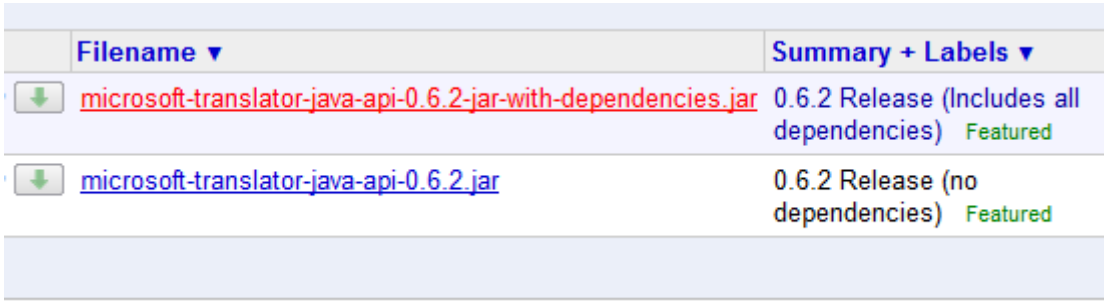
Για αυτό το project χρησιμοποιούμε το περιβάλλον NetBeans και δημιουργούμε ένα νέο project τύπου Java Application.



Κατά τη δημιουργία δε χρειάζεται καμία άλλη ιδιαίτερη ρύθμιση. Έπειτα πρέπει να εισάγουμε τα αρχεία της βιβλιοθήκης στο project, τα οποία βρίσκουμε από τη διεύθυνση:

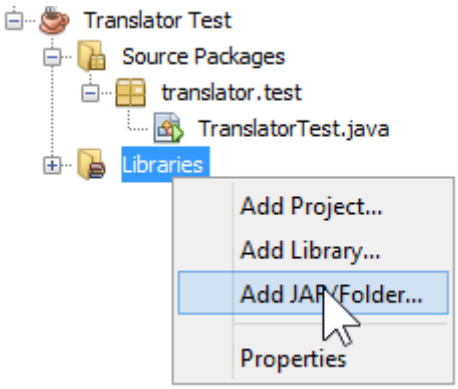
https://code.google.com/p/microsoft-translator-java-api/

Έχουμε δύο επιλογές για τη σελίδα, η μία είναι να κατεβάσουμε μόνο τα αρχεία της βιβλιοθήκης και η δεύτερη είναι να κατεβάζουμε και τα επιπλέον αρχεία που χρειάζονται για τη λειτουργία της. Επιλέγουμε τη δεύτερη, δηλαδή κατεβάζουμε το microsoft-translator-java-api-0.6.2-jar-with-dependencies.jar.

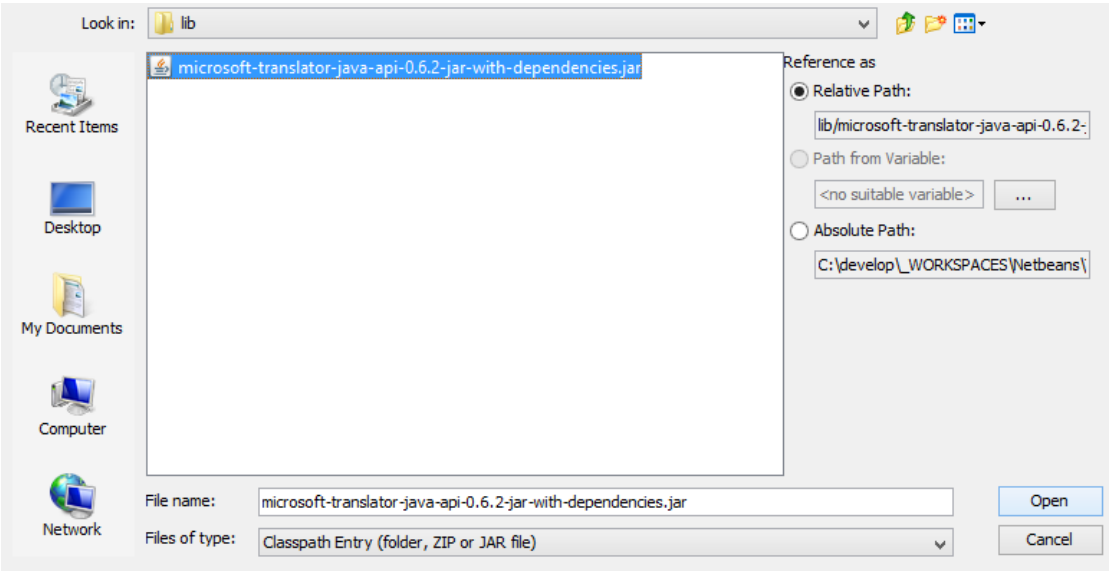


Μέσα στο φάκελο του project δημιουργούμε έναν φάκελο lib και γράφουμε το αρχείο που κατεβάσαμε. Μετά πρέπει να πούμε στο NetBeans ότι θέλουμε να χρησιμοποιήσουμε αυτό το αρχείο βιβλιοθήκης.

Για αυτό το σκοπό κάνουμε δεξί κλικ στο φάκελο Libraries που υπάρχει μέσα στο project και επιλέγουμε Add Folder/Jar.



Στο επόμενο παράθυρο επιλέγουμε το αρχείο JAR μέσα από τον φάκελο lib που δημιουργήσαμε προηγουμένως.



Μετά χρειάζεται να γράψουμε τον κώδικα για να ελέγξουμε τη λειτουργία μετάφρασης.

Ο κώδικας για αυτή την εφαρμογή είναι αρκετά απλός, καθώς δεν πρόκειται να ασχοληθούμε καθόλου με χειρισμό λαθών.

Αρχικά εισάγουμε το id της εφαρμογής και τον κωδικό πρόσβασης που έχουμε πάρει από την εγγραφή σας στο Translation API.

```
Translate.setClientId("OCR_SERVER");
Translate
   .setClientSecret("oCb8El1QYc3ta6QKTawyOidKs1CGE/vVm7Y9oMwQPh8=");
```

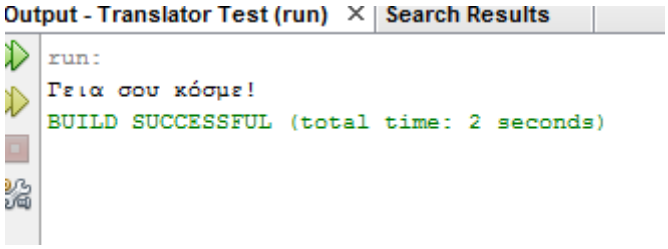
Στη συνέχεια ζητάμε από την εφαρμογή να μας μεταφράσει ένα απλό String από τα αγγλικά στα ελληνικά.

```
String translatedText = Translate
      .execute("Hello world!", Language.ENGLISH, Language.GREEK);
```

Μετά τυπώνουμε το String που επιστρέφει η μέθοδος execute.

```
System.out.println(translatedText);
```

Η έξοδος του προγράμματος είναι το παρακάτω:



Βλέπουμε ότι η μετάφραση έγινε με επιτυχία.

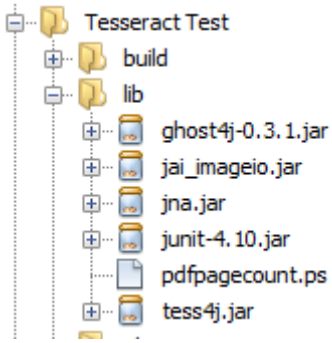
4.2.2 Αναγνώριση χαρακτήρων με το Tesseract

Για την οπτική αναγνώριση χαρακτήρων χρησιμοποιήσαμε τη βιβλιοθήκη Tesseract, η οποία είναι γραμμένη σε C++. Ο λόγος που χρησιμοποιούμε μία βιβλιοθήκη που δεν είναι γραμμένη σε Java είναι πως δεν υπάρχουν αντίστοιχες ικανοποιητικές λύσεις που να είναι γραμμένες σε Java, και από τις υπόλοιπες το Tesseract είναι το πιο ευρέως διαδεδομένο, εξαιτίας της ακρίβειας των αποτελεσμάτων του.

Για τη χρήση του είχαμε δύο επιλογές. Η πρώτη ήταν να το ενσωματώσουμε στην εφαρμογή android που θα γράψουμε και η δεύτερη να το τρέχουμε σε έναν εξωτερικό server. Επιλέξαμε τη δεύτερη επιλογή γιατί το OCR είναι μία εργασία ιδιαίτερα απαιτητική σε μνήμη και υπολογιστική ισχύ, και ήταν πιθανόν κάποιες συσκευές τηλεφώνου να μην μπορούν να ανταποκριθούν.

Για την επικοινωνία της βιβλιοθήκης με το πρόγραμμα Java που γράφουμε χρησιμοποιούμε τη βιβλιοθήκη Tess4J, που περιέχει κάποιες κλάσεις Java οι οποίες εσωτερικά επικοινωνούν με τα dll που παράγονται από τη C++ που είναι γραμμένο το Tesseract.

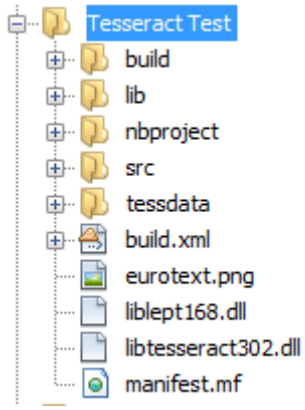
Αρχικά χρειάζεται να φτιάξουμε πάλι έναν φάκελο lib στο πρόγραμμά μας. Εκεί αντιγράφουμε τις βιβλιοθήκες του Tess4J. Αυτή τη φορά δεν έχουμε μόνο ένα αρχείο JAR, αλλά χρειάζεται να γράψουμε αρκετά από αυτά, όπως φαίνεται παρακάτω:



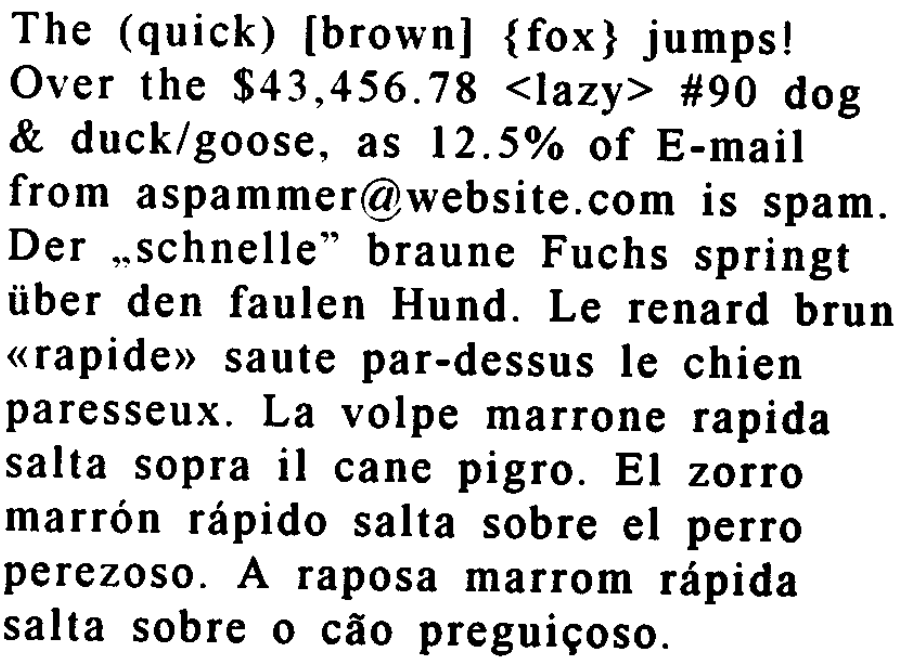
Προσοχή χρειάζεται και στο ότι περιλαμβάνουμε το αρχείο pdfpagecount.ps, το οποίο δεν είναι JAR.

Έπειτα πρέπει να γράψουμε τον φάκελο tessdata μέσα στον φάκελο του project. Αυτός περιέχει τα αρχεία εκπαίδευσης του προγράμματος, ώστε να αναγνωρίζει χαρακτήρες για διάφορες γλώσσες.

Επίσης πρέπει να γράψουμε και δύο αρχεία στο βασικό φάκελο του project. Αυτά είναι τα αρχεία liblept168.dll και libtesseract302.dll. Τα αρχεία dll είναι αρχεία βιβλιοθήκης, δηλαδή είναι εκτελέσιμα αρχεία για το λειτουργικό σύστημα Windows, τα οποία περιέχουν τον μεταγλωττισμένο κώδικα του προγράμματος. Σε λειτουργικό σύστημα linux προφανώς θα χρειαζόντουσαν διαφορετικές βιβλιοθήκες.



Επιπλέον σε αυτόν τον φάκελο βάζουμε και μία δοκιμαστική εικόνα, την eurotext.png, την οποία θα χρησιμοποιήσουμε για να δοκιμάσουμε τα αποτελέσματα. Τα περιεχόμενα της εικόνας φαίνονται παρακάτω.



Τα παραπάνω αρχεία χρειάζεται να βρίσκονται στον βασικό φάκελο του project.

Έπειτα γράφουμε το μικρό παράδειγμά μας για να δούμε αν όλα λειτουργούν κανονικά. Το πρώτο πράγμα που πρέπει να γίνει είναι να πούμε στη Java να αναλύσει τα αρχεία jar που έχουμε συμπεριλάβει για νέες δυνατότητες που περιέχουν.

```
ImageIO.scanForPlugins();
```

Έπειτα δημιουργούμε ένα στιγμιότυπο του αντικειμένου που αναπαριστά τη βιβλιοθήκη με την εντολή:

```
Tesseract instance = Tesseract.getInstance();
```

Μετά ανοίγουμε την εικόνα και την δίνουμε σαν παράμετρο στη βιβλιοθήκη για αναγνώριση. Το αποτέλεσμα επιστρέφεται άμεσα και το τυπώνουμε σαν String.

```
File imageFile = new File("eurotext.png");
String result = instance.doOCR(imageFile);
System.out.println(result);
```

Η έξοδος του προγράμματος είναι το παρακάτω κείμενο:

```
run:
The (quick) [brown] {fox} jumps!
Over the $43,456.78 <lazy> #90 dog
& duck/goose, as 12.5% of E-mail
from aspammer@website.com is spam.
Der ,,schnelle" braune Fuchs springt
iiber den faulen Hund. Le renard brun
arapide» saute par-dessus le chien
paresseux. La volpe marrone rapida
salta sopra i] cane pigro. El zorro
marrén répido salta sobre el perro
perezoso. A raposa marrom rzipida
salta sobre o e50 preguieoso.


BUILD SUCCESSFUL (total time: 1 second)
```

Το αποτέλεσμα δεν είναι το ίδιο με την εικόνα που δώσαμε αλλά έχει αποκλίσεις. Αυτό οφείλεται στο ότι η αρχική εικόνα είχε μία μικρή κλίση στα γράμματα και επιπλέον δεν ήταν απόλυτα καθαρά. Επίσης περιέχει χαρακτήρες από πάρα πολλές γλώσσες που είναι εύκολο να μπερδευτούν μεταξύ τους.

4.3 Πρωτόκολλο επικοινωνίας

Σε αυτή την ενότητα θα αναλύσουμε τη μορφή που πρέπει να έχουν τα δεδομένα που στέλνονται ανάμεσα στα μέρη του προγράμματος ώστε να μπορούν να επικοινωνήσουν μεταξύ τους.

4.3.1 Αποστολή δεδομένων

Η εφαρμογή που έχουμε γράψει για τον web server λειτουργεί σαν ένα REST web service, δηλαδή λαμβάνει δεδομένα με μορφή HTML.

Συγκεκριμένα με τη χρήση του πρωτοκόλλου HTTP πρέπει να σταλεί μία εικόνα με τη μορφή multipart/form-data. Το όνομα του πεδίου δεν έχει σημασία, εξαιτίας του τρόπου με τον οποίο παίρνουμε τα δεδομένα.

4.3.2 Μορφή απάντησης

Η απάντηση που στέλνουμε έχει τη μορφή JSON. Το JSON επιλέχθηκε για την ευκολία χρήσης και την ευελιξία που παρέχει, καθώς δεν χρειάζεται να ορίσουμε ένα νέο πρωτόκολλο από την αρχή αλλά χρησιμοποιούμε το JSON με τις κατάλληλες βιβλιοθήκες.

Η απάντηση που στέλνεται είναι η παρακάτω:

{

"originalText": "Optical character recognition, usually abbreviated to OCR, is\nthe mechanical or electronic conversion of scanned images of\nhandwritten, typewritten or printed text into machine-encoded\ntext. lt is widely used as a form of data entry from some sort of\noriginal paper data source, whether documents, sales receipts,\nmail, or any number of printed records. lt is a common method\nof digitizing printed texts so that they can be electronically\n\n",

"originalLanguage": "en",

"translatedText": "Οπτική αναγνώριση χαρακτήρων, συνήθως με τα αρχικά OCR, είναι η μηχανική ή ηλεκτρονική μετατροπή σαρωμένες εικόνες του χειρόγραφου, δακτυλογραφημένου ή τυπωμένου κειμένου στο μηχανή-κωδικοποιημένο κείμενο. lt χρησιμοποιείται ευρέως ως μια μορφή της εισαγωγής δεδομένων από κάποιο είδος του αρχικού αρχείου προέλευσης δεδομένων χαρτί, είτε έγγραφα, παραλαβές πωλήσεων, ταχυδρομείο, ή οποιοδήποτε αριθμό τυπωμένο εγγραφές. lt είναι μια κοινή μέθοδος της ψηφιοποίησης τυπωμένα κείμενα, έτσι ώστε να μπορούν να είναι ηλεκτρονικά\n\n"

}

Περιέχει 3 πεδία με τις εξής πληροφορίες:

- Το πεδίο **originalText** περιέχει σαν string το κείμενο που αναγνώρισε το Tesseract.
- Το πεδίο **originalLanguage** περιέχει τον κωδικό της γλώσσας στην οποία ανιχνεύτηκε ότι είναι το κείμενο της εικόνας.
- Το πεδίο **translatedText** είναι το μεταφρασμένο κείμενο, όπως επιστράφηκε από το Microsoft translate.

4.4 Εφαρμογή για το web

Η πρώτη εφαρμογή που θα αναλύσουμε είναι το web app, δηλαδή μία εφαρμογή που τρέχει σε έναν server Apache Tomcat και εκτελείται μέσω διαδικτύου. Ιδιαίτερη σημασία παίζουν τόσο ο κώδικας όσο και οι ρυθμίσεις που πρέπει να γίνουν ώστε να συνεργαστεί με τη βιβλιοθήκη Tesseract.

Η μεγαλύτερη δυσκολία που συναντήσαμε στην συγγραφή του web app δεν ήταν ο ίδιος ο κώδικας αλλά η ορθή ρύθμιση του project ώστε να μπορεί να εκτελέσει τις συναρτήσεις του Tesseract. Ο λόγος είναι, πως σε αντίθεση με το απλό project που είχαμε δημιουργήσει παραπάνω, εδώ δεν εκτελείται άμεσα το πρόγραμμά μας αλλά τρέχει μέσω του Tomcat που τρέχει μέσω του eclipse.

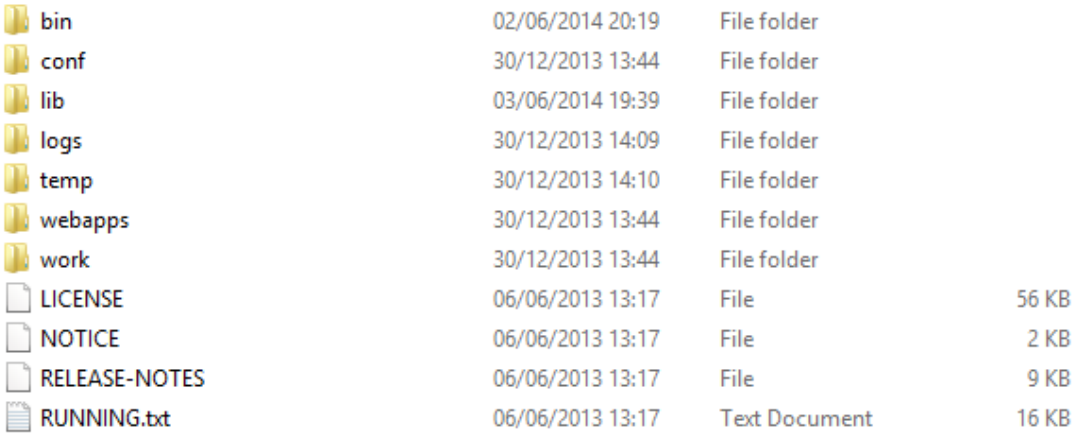
Αυτό το project γράφτηκε στο eclipse γιατί στάθηκε αδύνατο να ρυθμίσουμε το Netbeans σωστά για να λύσουμε το πρόβλημα. Η παντελής έλλειψη documentation για αυτή τη διαδικασία ήταν ένα πολύ μεγάλο εμπόδιο, και οδήγησε στο να κάνουμε μια σειρά πολύ χρονοβόρων πειραμάτων μέχρι να βρούμε τις σωστές ρυθμίσεις.

4.4.1 Ρύθμιση server

Πριν ξεκινήσουμε χρειάζεται να φτιάξουμε έναν νέο server. Για αυτή τη διαδικασία πρέπει να κατεβάσουμε τον Apache Tomcat όπως αναφέραμε παραπάνω.

Το αρχείο που κατεβάσαμε είναι ένα συμπιεσμένο αρχείο που περιέχει τα αρχεία της εγκατάστασης. Οι υπόλοιπες επιλογές διαφέρουν στον τρόπο εγκατάστασης, όπως αφού τον δικό μας server θα τον διαχειρίζεται το eclipse δεν υπάρχει λόγος να επιλέξουμε κάτι παραπάνω.

Αποσυμπιέζουμε αυτό το αρχείο σε έναν φάκελο στο δίσκο μας.

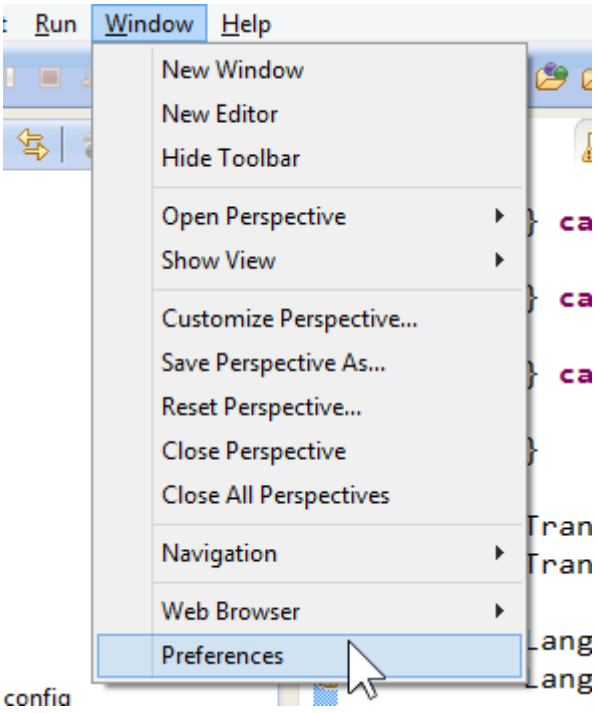


Όπως ανακαλύψαμε μετά από πολλές δοκιμές για να μπορεί ο Tomcat να δει τα εκτελέσιμα του Tesseract είναι απαραίτητο να τα γράψουμε μέσα στον φάκελο lib.

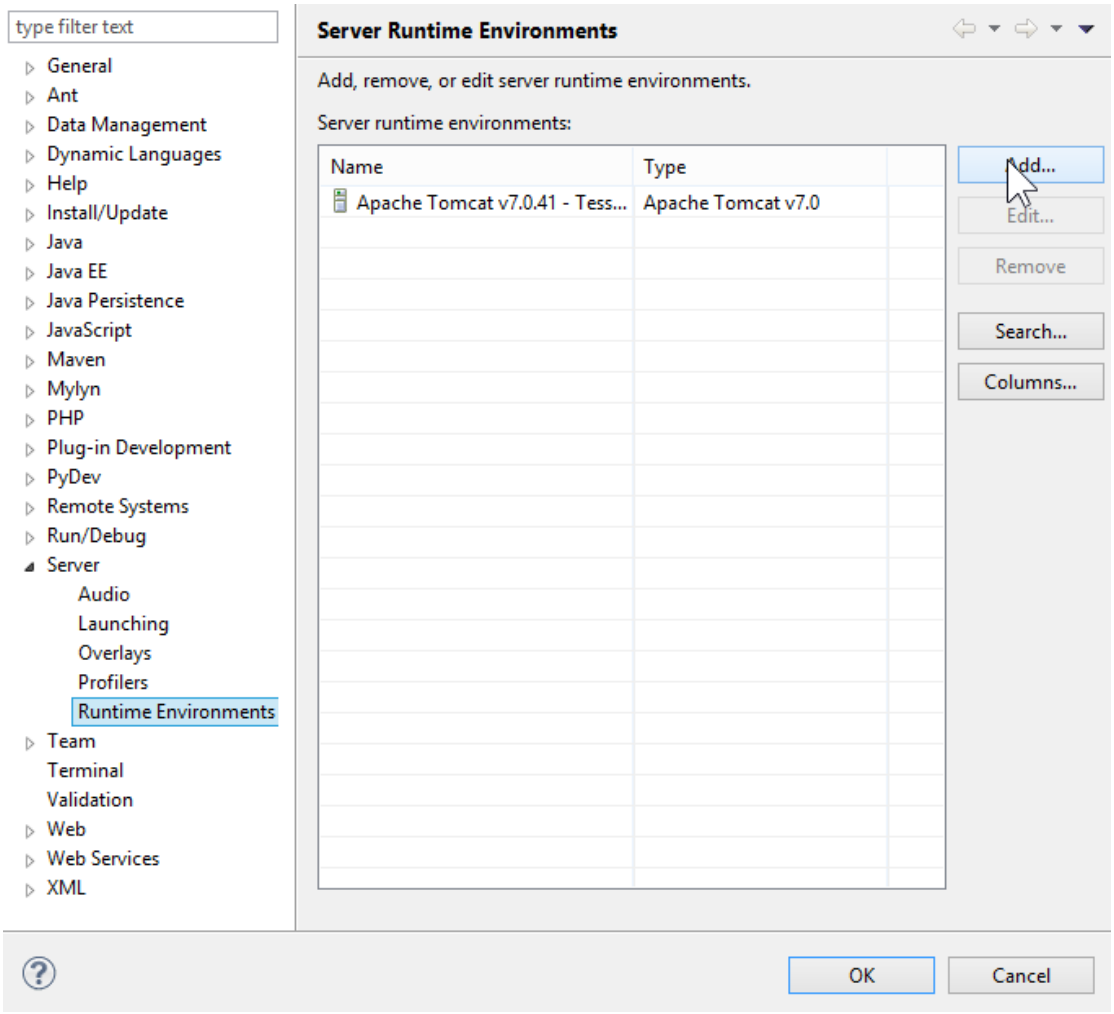
annotations-api.jar	06/06/2013 13:17	Executable Jar File	15 KB
catalina.jar	06/06/2013 13:17	Executable Jar File	1.538 KB
catalina-ant.jar	06/06/2013 13:17	Executable Jar File	53 KB
catalina-ha.jar	06/06/2013 13:17	Executable Jar File	132 KB
catalina-tribes.jar	06/06/2013 13:17	Executable Jar File	250 KB
ecj-4.2.2.jar	06/06/2013 13:17	Executable Jar File	1.760 KB
el-api.jar	06/06/2013 13:17	Executable Jar File	46 KB
jasper.jar	06/06/2013 13:17	Executable Jar File	586 KB
jasper-el.jar	06/06/2013 13:17	Executable Jar File	121 KB
jsp-api.jar	06/06/2013 13:17	Executable Jar File	87 KB
liblept168.dll	28/02/2013 21:17	Application extens...	1.633 KB
libtesseract302.dll	03/08/2013 12:43	Application extens...	1.538 KB
servlet-api.jar	06/06/2013 13:17	Executable Jar File	174 KB
tomcat-api.jar	06/06/2013 13:17	Executable Jar File	7 KB
tomcat-coyote.jar	06/06/2013 13:17	Executable Jar File	778 KB
tomcat-dbcp.jar	06/06/2013 13:17	Executable Jar File	230 KB
tomcat-i18n-es.jar	06/06/2013 13:17	Executable Jar File	76 KB
tomcat-i18n-fr.jar	06/06/2013 13:17	Executable Jar File	48 KB
tomcat-i18n-ja.jar	06/06/2013 13:17	Executable Jar File	51 KB
tomcat-jdbc.jar	06/06/2013 13:17	Executable Jar File	122 KB
tomcat-util.jar	06/06/2013 13:17	Executable Jar File	23 KB

Έπειτα από τις επιλογές του eclipse πρέπει να ορίσουμε τον νέο server. Αυτό σημαίνει πώς πρέπει να δηλώσουμε που βρίσκεται ο νέος server ώστε να τον χρησιμοποιούμε έπειτα για να τρέξουμε το project μας.

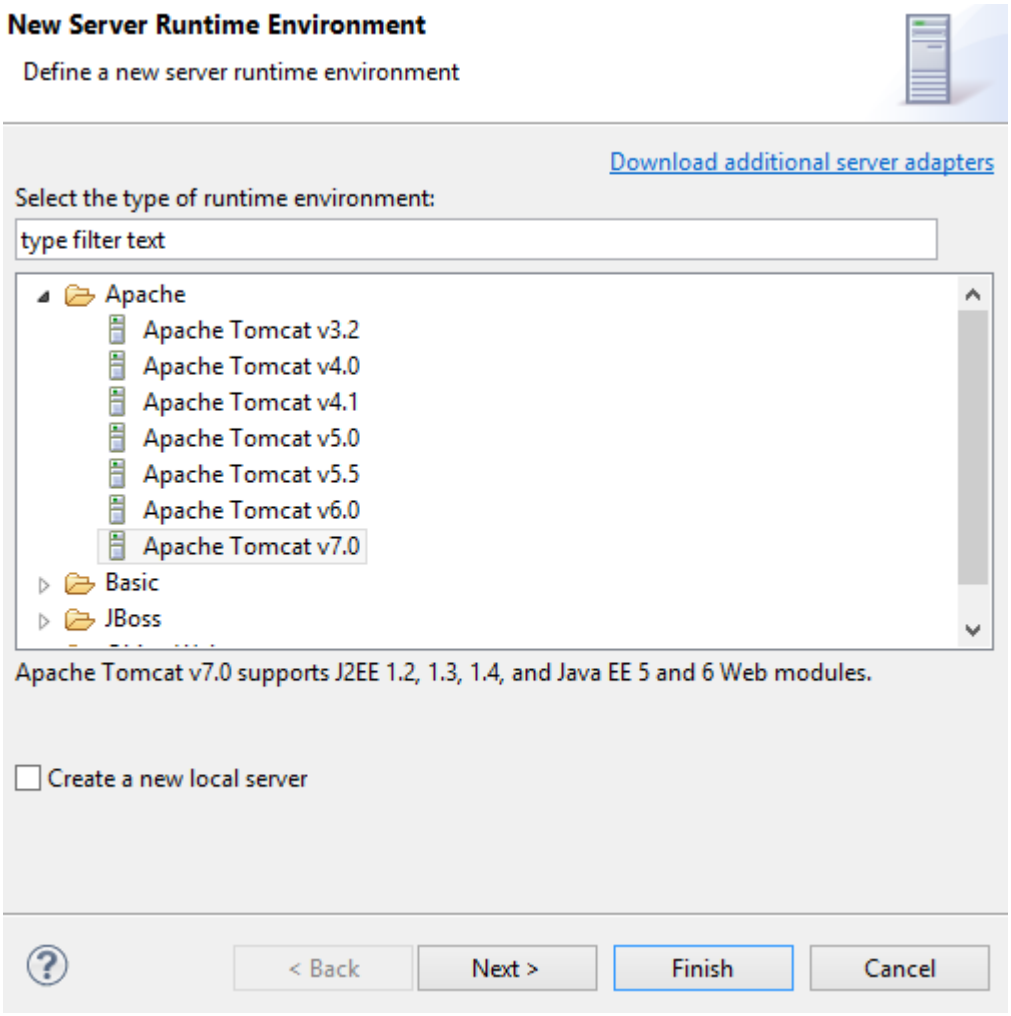
Αυτό γίνεται από το μενού Window → Preferences.



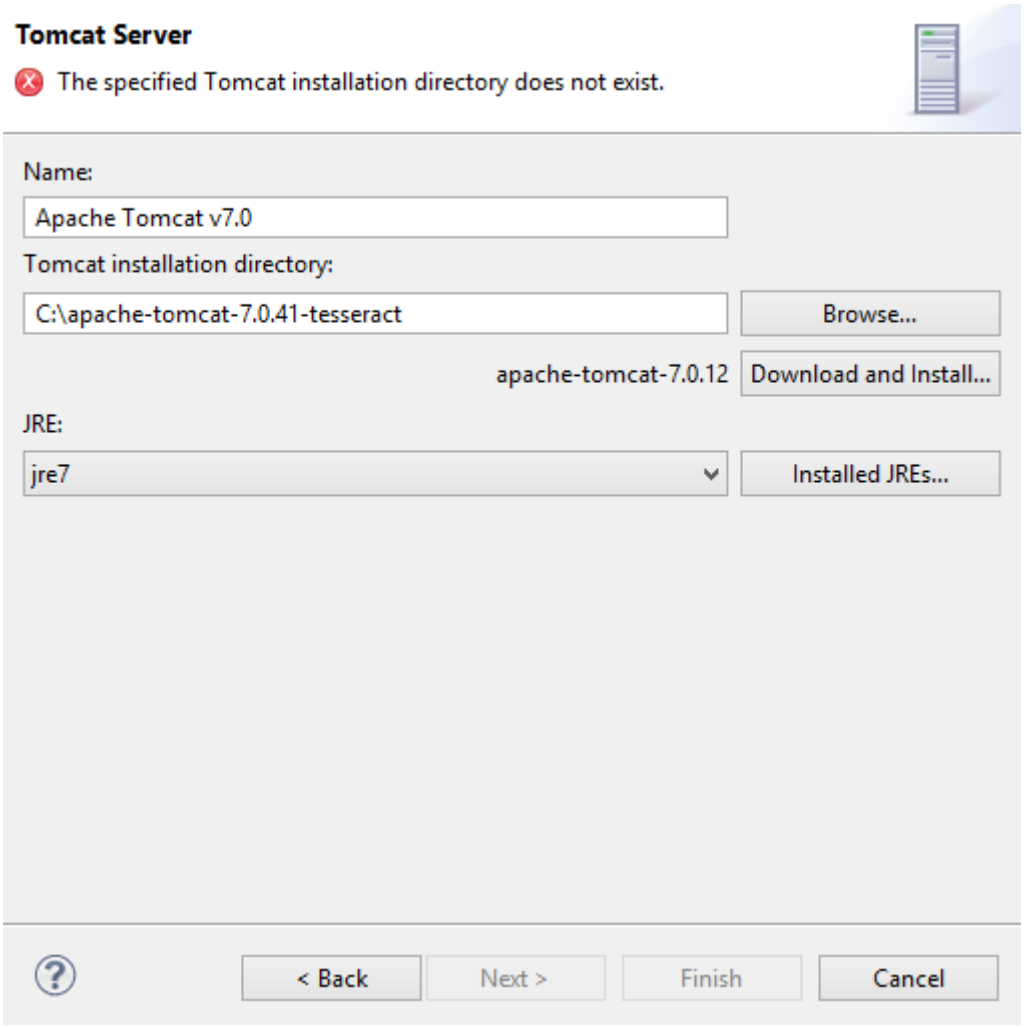
Εκεί φαίνεται η λίστα με τις επιλογές του eclipse. Επιλέγουμε το Server → Runtime Environments και πατάμε το κουμπί Add.



Από το μενού επιλέγουμε το Apache → Apache Tomcat 7.0. Τη στιγμή που γράφαμε το project το eclipse υποστήριζε τον Tomcat 7, αν και η τελευταία έκδοση του Tomcat είναι η 8, οπότε χρησιμοποιήσαμε την παλιότερη έκδοση.



Στο επόμενο παράθυρο χρειάζεται να ορίσουμε τον φάκελο της εγκατάστασης του Tomcat. Εκεί βάζουμε τον βασικό φάκελο και όχι τον bin ή το lib.

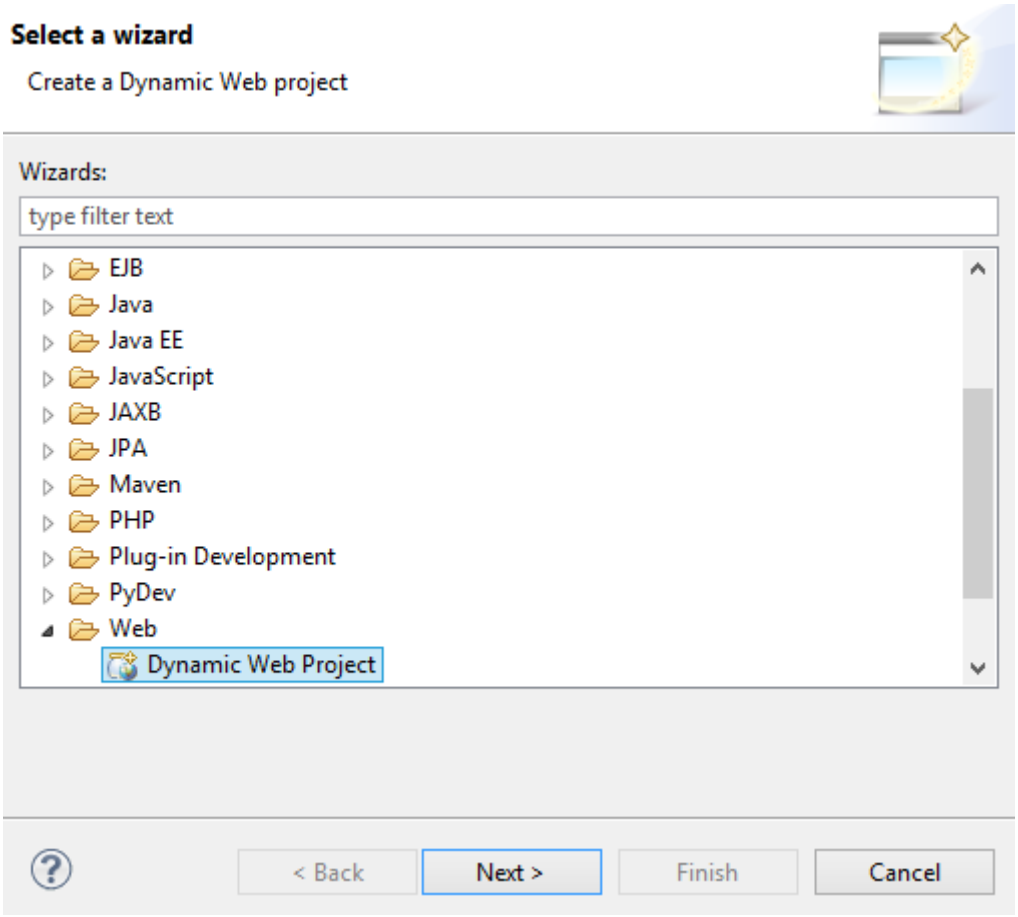


Χρειάζεται μεγάλη προσοχή στην επιλογή του JRE. Το JRE είναι η Virtual Machine που χρησιμοποιεί το eclipse για να τρέξει τον Tomcat. Αν έχουμε ένα σύστημα 64 bit έχουμε την επιλογή να χρησιμοποιήσουμε την αντίστοιχη έκδοση της Java. Όμως τα εκτελέσιμα του Tesseract έχουν γίνει compile για αρχιτεκτονική 32 bit και δεν μπορούν να τρέξουν με 64μπιτη έκδοση της Java.

Όταν τρέχει το eclipse χρησιμοποιεί αυτόματα το Default JRE, το οποίο είναι και η τυπική επιλογή όταν δημιουργούμε νέο server. Αν έχουμε εγκαταστήσει πάνω από μία έκδοση java είναι απαραίτητο να ορίσουμε στο παραπάνω μενού το JRE με αρχιτεκτονική 32 bit.

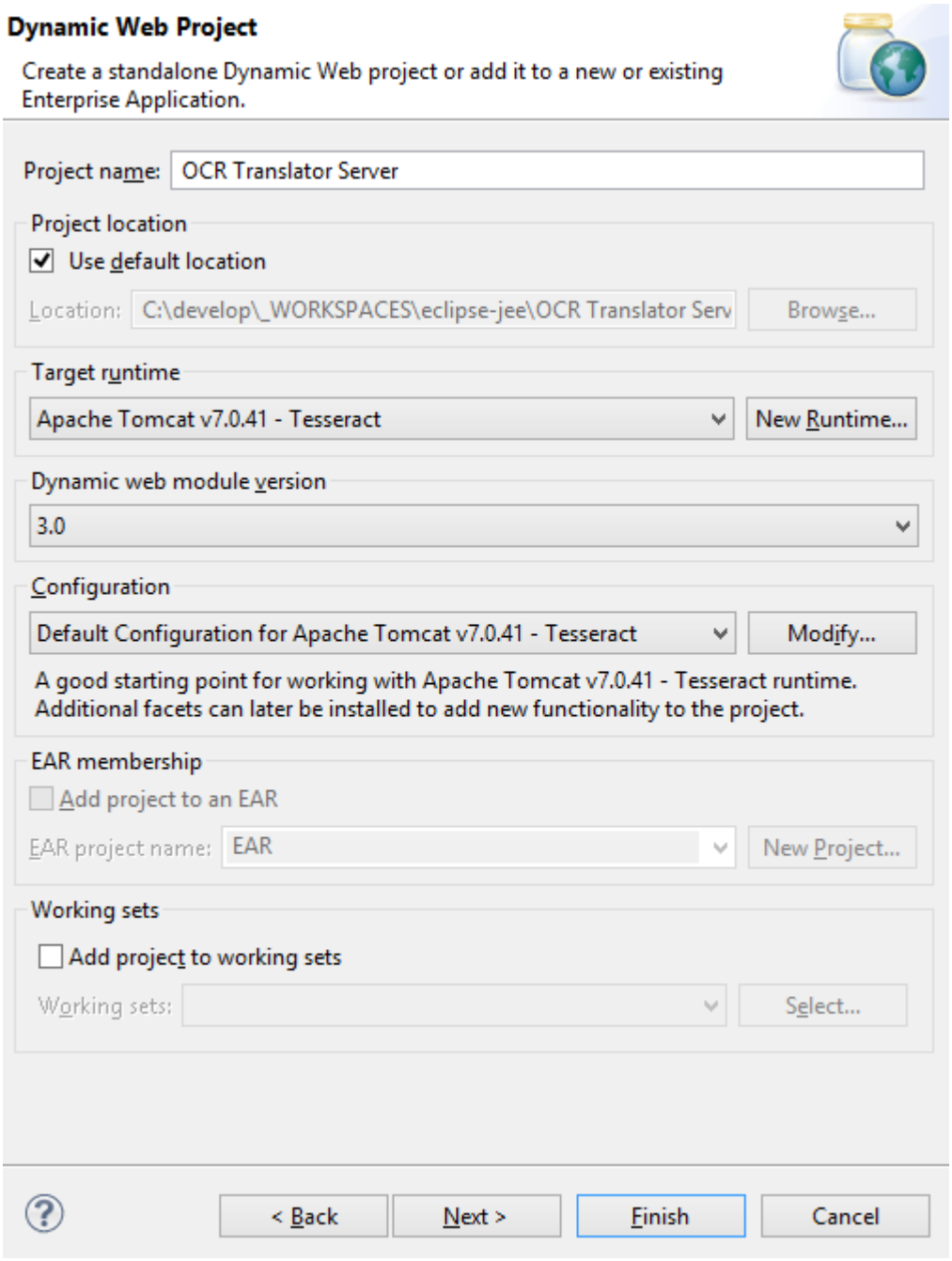
4.4.2 Ρυθμίσεις project

Για να ξεκινήσουμε το project φτιάχνουμε ένα νέο project στο eclipse επιλέγοντας File → New → Project… και από τη λίστα διαλέγουμε την επιλογή Web → Dynamic web project.

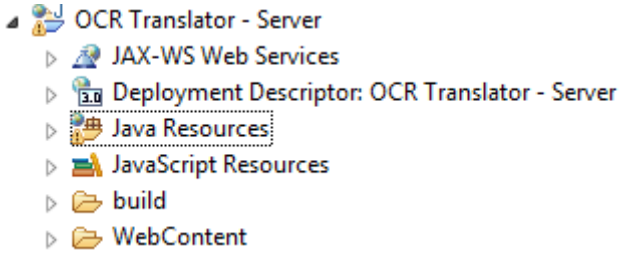


Στην επόμενη οθόνη επιλέγουμε το όνομα του project και τον server που έχουμε ορίσει.

Οι υπόλοιπες ρυθμίσεις δεν μας ενδιαφέρουν αυτή τη στιγμή και αρκεί να πατήσουμε το Finish για να κατασκευαστεί το νέο project.



Αυτό φτιάχνει το νέο project. Η μορφή των φακέλων του φαίνεται στην παρακάτω εικόνα.



Για την εκτέλεσή του χρειάζεται ένας αριθμός από αρχεία jar, συγκεκριμένα αυτά που είδαμε στην δοκιμαστική επιλογή για το Tesseract. Αυτά πρέπει να αντιγραφούν στον φάκελο του project, και συγκεκριμένα στο WebContent/WEB-INF/lib.

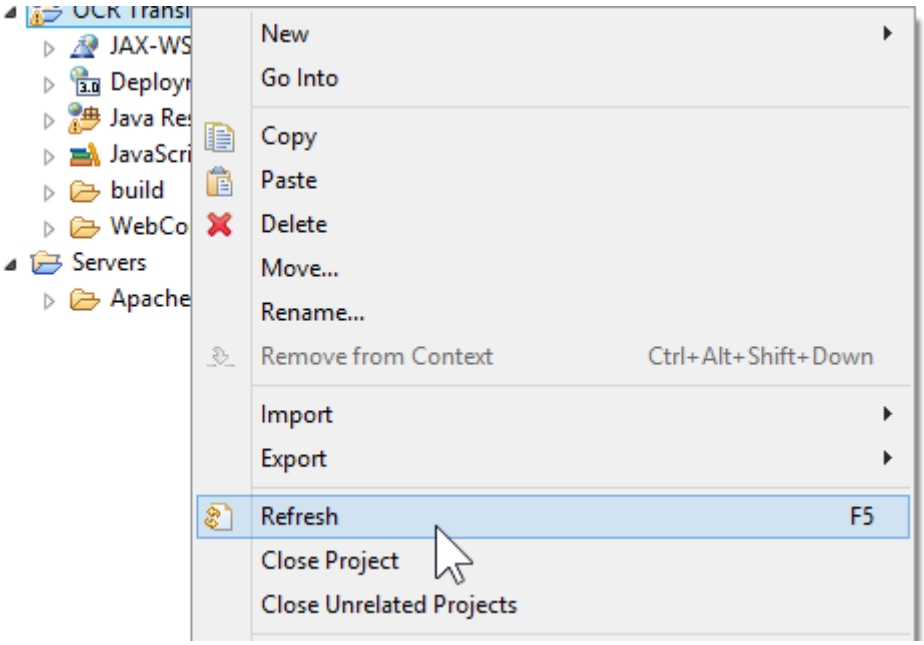
tessdata	03/06/2014 20:20	File folder	
commons-fileupload-1.2.2.jar	15/07/2010 00:43	Executable Jar File	59 KB
commons-io-2.4.jar	13/06/2012 01:19	Executable Jar File	181 KB
ghost4j-0.3.1.jar	28/02/2013 21:17	Executable Jar File	31 KB
jai_imageio.jar	28/02/2013 21:17	Executable Jar File	1.114 KB
jna.jar	28/07/2013 13:21	Executable Jar File	894 KB
json-simple-1.1.1.jar	05/10/2012 18:22	Executable Jar File	24 KB
microsoft-translator-java-api-0.6.1-jar-wi...	05/10/2012 18:04	Executable Jar File	36 KB
pdfpagecount.ps	28/02/2013 21:17	PS File	1 KB
tess4j.jar	22/09/2013 11:07	Executable Jar File	68 KB

Εκεί γράφουμε όλα τα αρχεία jar που χρησιμοποιήσαμε προηγουμένως. Αυτά είναι το jar για το Microsoft Translator, το json-simple καθώς και τα jar που χρειαστήκανε για το Tesseract.

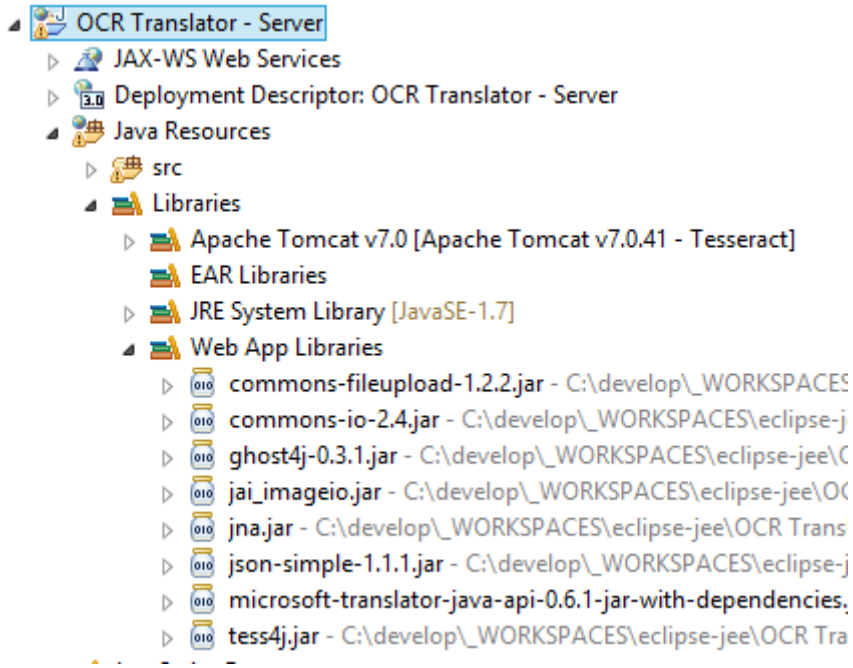
Χρειάζεται προσοχή στο ότι πέρα από τα αρχεία jar γράψαμε τον φάκελο tessdata και το αρχείο pdfpagecount.ps. Αυτά είναι απαραίτητα για να μην πετάει λάθη όταν τρέχει το project.

Θεωρητικά το eclipse χρησιμοποιεί αυτόματα σα βιβλιοθήκες όλα τα αρχεία που βρίσκονται σε αυτόν τον φάκελο για το project, όμως η εμπειρία μας έχει δείξει πώς κάτι τέτοιο δε συμβαίνει πάντα.

Για να δούμε αν έχει καταλάβει την ύπαρξή τους αρχικά κάνουμε δεξί κλικ στο project και πατάμε την επιλογή Refresh.

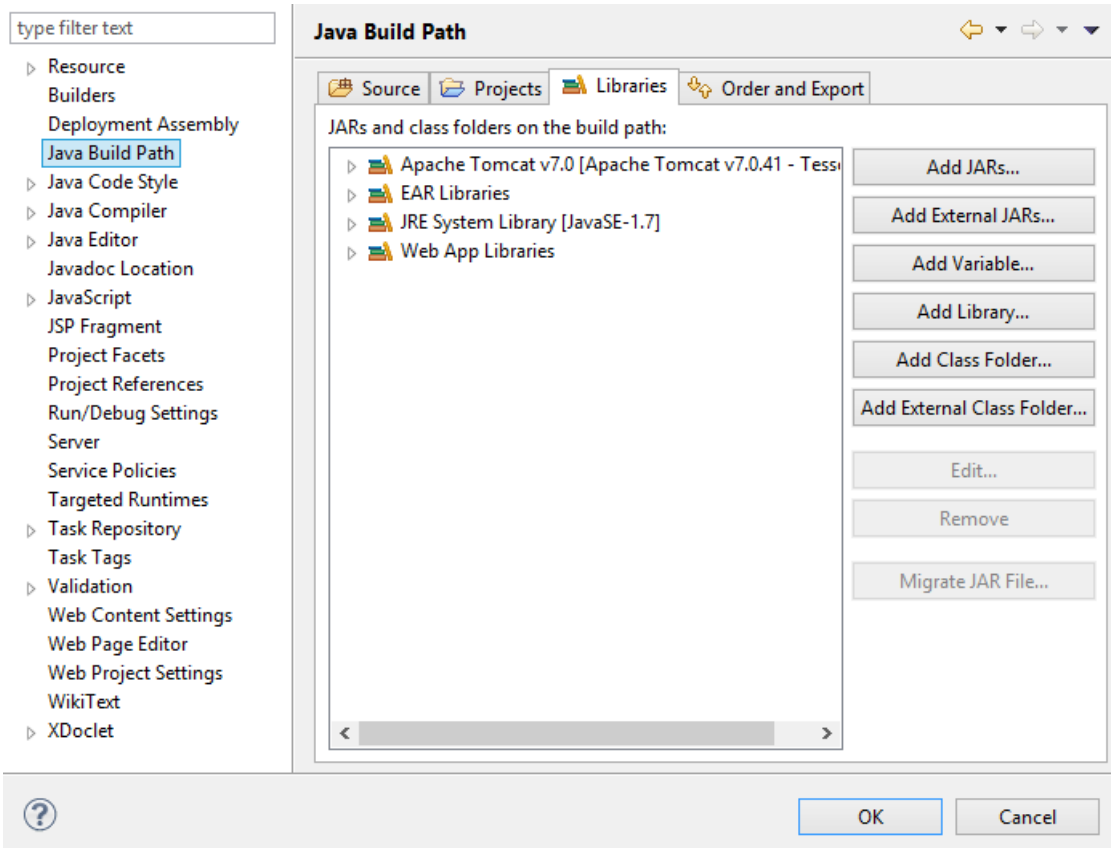


Έπειτα στο δέντρο που δείχνει τα περιεχόμενα του project ανοίγουμε το Libraries → Web App Libraries. Εκεί ελέγχουμε αν φαίνονται τα αρχεία που γράψαμε, θα πρέπει να εμφανίζονται όπως παρακάτω.



Αν αυτό δε γίνεται, κάτι που συναντήσαμε αρκετά συχνά, πρέπει να τα ορίσουμε εμείς. Χρειάζεται να κάνουμε δεξί κλικ στο project και μετά να επιλέξουμε το properties.

Από το παράθυρο που φαίνεται διαλέγουμε το Java Build Path και πατάμε το κουμπί Add JARs.



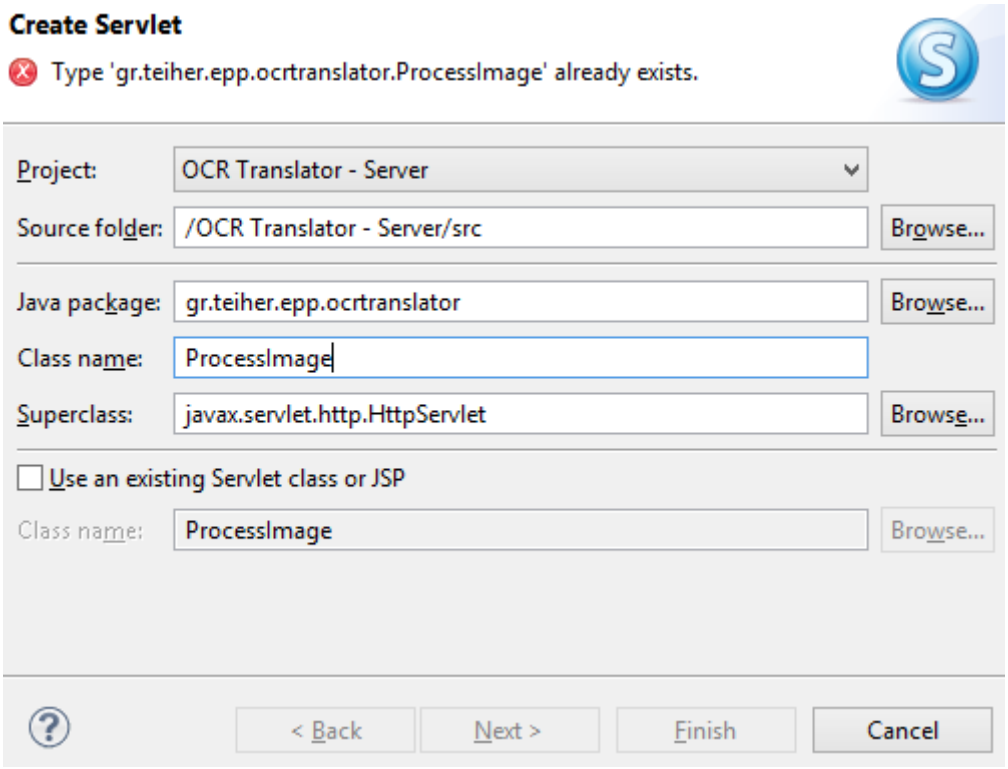
Από το επόμενο παράθυρο επιλέγουμε μέσα από το φάκελο του project στη διαδρομή WebContent → WEB_INF → lib όλα τα αρχεία jar που εμφανίζει και πατάμε OK.

4.4.3 Επεξήγηση κώδικα

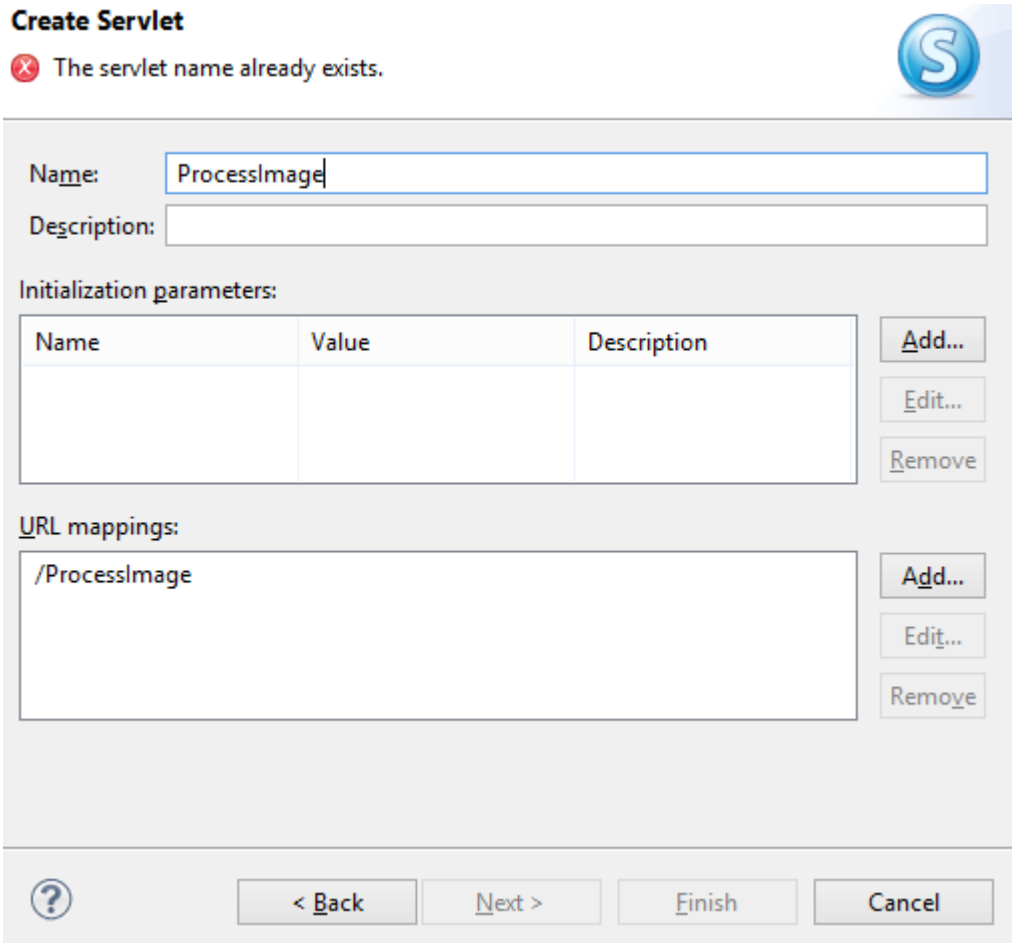
4.4.3.1 Servlet ProcessImage

Ο κώδικας του project είναι συγκεντρωμένος σε ένα servlet, το ProcessImage. Αυτό το γράψαμε στο πακέτο gr.teiher.epp.ocrtranslator. Αρχικά φτιάχνουμε το νέο πακέτο κάνοντας δεξί κλικ στο Java Resourses / src και επιλέγοντας New → Java package, και σαν όνομα δίνουμε το παραπάνω.

Μετά κάνουμε δεξί κλικ πάνω στο πακέτο που μόλις φτιάξαμε και διαλέγουμε New → Servlet. Στο παράθυρο που εμφανίζεται δίνουμε το όνομα του servlet που θα γράψουμε.



Μετά πατάμε το Next. Σε αυτό το παράθυρο πρέπει να δοθεί προσοχή στο πεδίο URL Mappings. Αυτό έχει τη διαδρομή με την οποία θα μπορούμε να το καλούμε στον Server. Η αρχική τιμή είναι ίδια με το όνομα του servlet, όμως αν δεν μας κάνει μπορούμε να δώσουμε κάτι άλλο.



Μετά γράφουμε τον κώδικα. Όταν δημιουργούμε ένα νέο servlet εμφανίζονται κάποιες έτοιμες μέθοδοι.

Η πρώτη είναι ο default constructor.

```
public ProcessImage() {
	super();

}
```

Σε αυτόν δεν χρειάζεται να κάνουμε κάτι. Αν για κάποιο λόγο θέλαμε να κάνουμε κάποιες λειτουργίες κατά την αρχικοποίηση θα μπορούσαμε να γράψουμε τις κατάλληλες εντολές εδώ, όμως κάτι τέτοιο δεν είναι απαραίτητο για τις ανάγκες του αντικειμένου που γράφουμε.

Έπειτα υπάρχει η μέθοδος doGet η οποία χρησιμοποιείται για να επεξεργάζεται τις αιτήσεις GET.

```
	protected void doGet(HttpServletRequest request,
			HttpServletResponse response) throws
ServletException, IOException {

	}
```

Και σε αυτό το τμήμα του κώδικα δεν κάνουμε τίποτα. Το servlet μας παίρνει μία εικόνα που κάνει upload ο χρήστης. Αυτή μπορεί να μεταδοθεί μόνο μέσω του post, που έχουμε παρακάτω.

Η μέθοδος doPost είναι αυτή που επεξεργάζεται τις αιτήσεις POST, και αυτή που επιτελεί τις περισσότερες λειτουργίες.

```
	protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
```

Το πρώτο πράγμα που κάνουμε είναι να θέσουμε την κωδικοποίηση του αιτήματος και του αποτελέσματος σε UTF-8, ώστε να μπορούμε να στείλουμε πολυγλωσσικό κείμενο.

```
	request.setCharacterEncoding("UTF-8");
	response.setCharacterEncoding("UTF-8");
```

Έπειτα ορίζουμε τους headers της απάντησης για να έχουν τύπο JSON.

```
	response.setContentType("application/json");
```

Αυτό είναι βασικό να γίνει στην αρχή του κώδικα γιατί αν αρχίσουμε να γράφουμε στην απάντηση δεν μπορούμε μετά να αλλάξουμε τον τύπο της. Το ίδιο ισχύει και για τις εντολές της κωδικοποίησης γιατί, αν και την αλλάζουμε με εντολές Java, αυτές πρακτικά προσθέτους επιπλέον headers.

Επιπλέον ζητάμε από τη Java να διατρέξει τα αρχεία JAR που βάλαμε για να βρει νέους αποκωδικοποιητές για να ανοίγει επιπλέον μορφές εικόνων.

```
ImageIO.scanForPlugins();
```

Μετά παίρνουμε το stream της απάντησης και το αποθηκεύουμε στη μεταβλητή out. Αυτό είναι ένας PrintWriter, καθώς αυτά τα αντικείμενα ορίζονται από τις προδιαγραφές του servlet.

```
PrintWriter out = response.getWriter();
```

Το servlet μας πρέπει να παίρνει οπωσδήποτε μία εικόνα, και για αυτό το περιεχόμενο που του στέλνεται είναι απαραίτητο να είναι τύπου multipart.

Στην αρχή κάνουμε έλεγχο αν αυτό ισχύει, και αν δεν είναι κάνουμε return.

```
if (!ServletFileUpload.isMultipartContent(request)) {
    return;
}
```

Αν είναι όντως τύπου multipart τότε δημιουργούμε ένα αντικείμενο upload που είναι τύπου ServletFileUpload για να χειριστούμε την εικόνα που έδωσε ο χρήστης. Για να δημιουργηθεί αυτό πρέπει να έχει και ένα αντικείμενο τύπου FileItemFactory.

Η επόμενη μεταβλητή είναι ένα InputStream για να πάρουμε την εικόνα που έχει δοθεί.

```
InputStream inputStream = null;
```

Αυτό δεν μπορούμε να το αρχικοποιήσουμε απευθείας γιατί η αίτηση είναι τύπου multipart, δηλαδή περιέχει πολλά τμήματα και το κάθε ένα έχει το δικό του InputStream

Χρειαζόμαστε και δύο μεταβλητές τύπου String που περιέχουν τις default τιμές για το κείμενο που αναγνωρίζεται και μεταφράζεται.

```
String originalText = "OCR FAILED";
String translatedText = "TRANSLATION FAILED";
```

Η λογική είναι πώς αν όλα λειτουργήσουν ομαλά τα μηνύματα αποτυχίας θα αντικατασταθούν με τα σωστά δεδομένα. Όμως σε περίπτωση αποτυχίας δεν θα αλλάξει η τιμή τους και θα σταλεί το κατάλληλο μήνυμα λάθους.

Τέλος φτιάχνουμε το αντικείμενο JSON που περιέχει την απάντηση.

```
JSONObject obj = new JSONObject();
```

Στο επόμενο μέρος προσπαθούμε να διατρέξουμε τα δεδομένα, να πάρουμε την εικόνα και να τη δώσουμε στο Tesseract να την αναγνωρίσει.

Από εδώ και πέρα ο κώδικας μπορεί να πετάξει exceptions. Γι' αυτό το λόγο βάζουμε μεγάλο μέρος από το υπόλοιπο τμήμα σε ένα try – catch.

```
try {
      …
      …
      …
} catch (FileUploadException e) {
      System.err.println("Error during upload");
      e.printStackTrace();
} catch (TesseractException e) {
      System.err.println("OCR error");
      e.printStackTrace();
} catch (Exception e) {
      e.printStackTrace();
}
```

Αν συμβεί κάποιο από τα παραπάνω λάθη τυπώνουμε ένα βοηθητικό μήνυμα λάθος και μετά το stack trace του exception ώστε να αναλύσουμε που γίνεται το λάθος και να είναι πιο εύκολο να το διορθώσουμε.

Το πρώτο πράγμα που πρέπει να κάνουμε είναι να πάρουμε την εικόνα που έστειλε ο χρήστης. Για αυτό παίρνουμε μια λίστα με τα περιεχόμενα της αίτησης.

```
Iterator iterator = items.iterator();
```

Αυτή τη λίστα πρέπει να τη διατρέξουμε και να δούμε ποιο απ' όλα είναι η εικόνα. Για να τη διατρέξουμε παίρνουμε τον Iterator της.

Έπειτα με μία while διατρέχουμε τα αντικείμενα της λίστας.

```
while (iterator.hasNext()) {
      FileItem item = (FileItem) iterator.next();

      if (!item.isFormField()) {
            inputStream = item.getInputStream();
            break;
      }
}
```

Παίρνουμε τα αντικείμενα με τη σειρά από τον iterator με την εντολή iterator.next().

Για κάθε ένα από αυτά πρέπει να ελέγξουμε αν είναι πεδίο της φόρμας με το item.isFormField(). Ψάχνουμε να βρούμε την εικόνα, οπότε μας ενδιαφέρει το αντικείμενο που δεν είναι τέτοιου είδος πεδίου.

Όταν το βρούμε παίρνουμε το input stream της εικόνας και το αποθηκεύουμε στην μεταβλητή που φτιάξαμε παραπάνω. Μετά σταματάμε τη while με ένα break.

Έχοντας το inputStream της εικόνας μπορούμε να αρχικοποιήσουμε το Tesseract. Αυτό γίνεται με την εντολή:

```
Tesseract instance = Tesseract.getInstance();
```

Χρειάζεται προσοχή ότι για συστήματα 64 bit δεν μπορούμε να χρησιμοποιήσουμε το αντικείμενο Tesseract, αλλά πρέπει να χρησιμοποιήσουμε το παρακάτω:

```
Tesseract1 instance = new Tesseract1();
```

Στο περιβάλλον που εκτελείται η εφαρμογή, δηλαδή μέσα από τον Tomcat, η βιβλιοθήκη δεν μπορεί να ανιχνεύσει που βρίσκονται τα αρχεία δεδομένων της, και είναι απαραίτητο να προσδιορίσουμε τη διαδρομή. Ο προσδιορισμός της διαδρομής δεν μπορεί να γίνει χρησιμοποιώντας με απόλυτο path, καθώς αυτό αλλάζει κάθε φορά που το eclipse τρέχει τον Tomcat.

Για αυτό το λόγο παίρνουμε τη διεύθυνση με την παρακάτω εντολή.

```
String tessdataPath = 
getServletContext().getRealPath("/WEB-INF/lib/tessdata/");
```

Έπειτα δηλώνουμε αυτή τη διεύθυνση στο Tesseract.

```
instance.setDatapath(tessdataPath);
```

Μετά δημιουργούμε την εικόνα από το inputStream που κρατήσαμε.

```
BufferedImage image = ImageIO.read(inputStream);
```

Αυτή την εικόνα τη δίνουμε στο tesseract για να την αναλύσει και μετά αποθηκεύουμε το αποτέλεσμα στο String που έχουμε ορίσει.

```
originalText = instance.doOCR(image);
```

Έπειτα χρειάζεται να αρχικοποιήσουμε το Microsoft Translate ώστε να το χρησιμοποιήσουμε. Αυτό γίνεται περνόντας το client id και το secret key που έχουμε πάρει από την εγγραφή μας στην υπηρεσία.

```
Translate.setClientId("OCR_SERVER");
Translate.setClientSecret("oCb8El1QYc3ta6QKTawyOidKs1CGE/
vVm7Y9oMwQPh8=");
```

Έπειτα δίνουμε τη γλώσσα του αρχικού κειμένου και τη γλώσσα του αποτελέσματος.

```
Language originalLanguage = null;
Language transationLanguage = Language.GREEK;
```

Όταν κάνουμε τη μετάφραση έχουμε τη δυνατότητα να ορίσουμε την αρχική γλώσσα. Βάζοντας null ενεργοποιούμε την δυνατότητα αυτόματης ανίχνευσης της. Επιπλέον στο translation language επιλέγουμε ελληνικά.

Την αυτόματη ανίχνευση γλώσσας την χρησιμοποιούμε παρακάτω.

```
originalLanguage = Detect.execute(originalText);
```

Μετά μεταφράζουμε το κείμενο, δίνοντας σαν παραμέτρους το αρχικό κείμενο, την αρχική γλώσσα και τη γλώσσα στην οποία θέλουμε να μεταφραστεί.

```
translatedText = Translate.execute(originalText,
originalLanguage, Language.GREEK);
```

Αφού γίνει αυτό βάζουμε στο κείμενο της απάντησης τον κωδικό της γλώσσας προορισμού.

```
obj.put("originalLanguage",originalLanguage.toString());
```

Σε αυτό το σημείο τελειώνει ο κώδικας του exception που είχαμε προηγουμένως. Αυτό το οποίο έχει μείνει είναι να δημιουργήσουμε την απάντηση. Βάζουμε μέσα στα πεδία JSON originalText και translatedText τις τιμές των αντίστοιχων μεταβλητών.

```
obj.put("originalText", originalText);
obj.put("translatedText", translatedText);
```

Τέλος χρησιμοποιούμε τον PrintWriter που χρησιμοποιήσαμε προηγουμένως ώστε να γράψουμε τα αποτελέσματα.

```
obj.writeJSONString(out);
```

Από εκεί και πέρα η απάντηση που κατασκευάσαμε μέσα στον PrintWriter θα αποσταλεί αυτόματα στον πελάτη που έκανε το αντίστοιχο αίτημα.

4.4.3.2 Δοκιμαστικό αρχείο uploadForm.jsp

Για τη δοκιμή της εφαρμογής μας γράψαμε και απλή μία φόρμα. Είναι το αρχείο uploadForm.jsp, αν και χρησιμοποιεί μόνο στοιχεία HTML.

Καθώς το servlet μας δέχεται δεδομένα μέσω του πρωτοκόλλου HTTP, η φόρμα είναι απόλυτα συμβατή με αυτό και μπορεί να του στείλει τα δεδομένα με τη μορφή που περιμένει.

Η μορφή της φόρμας είναι:

```
<form method="post" action="ProcessImage"
enctype="multipart/form-data">
      <input type="file" name="imageFile" />

      <input type="submit" value="Επεξεργασία">
</form>
```

Χρειάζεται να δοθεί προσοχή στα σημεία που εξετάζουμε παρακάτω:

Σαν action δίνουμε το ProcessImage, που είναι το όνομα με το οποίο έχουμε δηλώσει σαν URL mapping για το servlet μας. Δεν δίνουμε καμία διαδρομή, ώστε να ζητηθεί από τη διαδρομή της φόρμας, που είναι στο ίδιο web app.

Στον πεδίο method είναι απαραίτητο να έχει τιμή post, καθώς είναι ο μοναδικός τρόπος με τον οποίο μπορούν να σταλούν αρχεία, δηλαδή η εικόνα που κάνει upload ο χρήστης.

Το πεδίο enctype=”multipart/form-data” είναι απαραίτητο ώστε η φόρμα να μπορεί να στείλει δεδομένα μέσω της λειτουργίας post.

Τέλος το πεδίο στο οποίο στέλνεται το αρχείο είναι τύπου file. Το όνομα του πεδίου δεν έχει σημασία, καθώς όπως φάνηκε στον κώδικα χειρισμού των δεδομένων στο servlet δεν το χρησιμοποιούμε για να ανιχνεύσουμε που βρίσκεται το αρχείο.

Για να τρέξουμε την εφαρμογή μέσα από το eclipse κάνουμε δεξί κλικ στο project και επιλέγουμε Run → Run on server. Την πρώτη φορά που το τρέχουμε μπορεί να μας

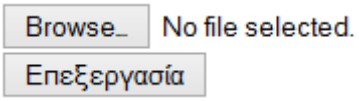
ζητηθεί να επιλέξουμε τον κατάλληλο server, όπου επιλέγουμε τον Tomcat που δημιουργήσαμε παραπάνω.

Το eclipse θα τρέξει και τον default browser του συστήματος, όμως θα εμφανιστεί ένα λάθος. Αυτό οφείλεται στο ότι το default index file πρέπει να ονομάζεται index.jsp. Αυτό δεν μας απασχολεί καθώς αυτό το τμήμα του project δε θα χρησιμοποιηθεί ποτέ πέρα από την προσωπική μας χρήση για τη δομική των δεδομένων.

Η διεύθυνση του αρχείου μας είναι η:

http://localhost:8080/OCR_Translator/uploadForm.jsp

Αυτό εμφανίζει την πολύ απλή φόρμα που κατασκευάσαμε.



Πατώντας Browse εμφανίζεται το στάνταρ παράθυρο του browser που μας ζητάει να επιλέγουμε ένα αρχείο. Μετά από αυτό το κουμπί επεξεργασία στέλνει τα δεδομένα στον server.

Το αποτέλεσμα εμφανίζεται με τη μορφή JSON.


```
{
    originalText: "Optical character recognition, usually abbreviated to
    OCR, is\nthe mechanical or electronic conversion of scanned images
    of\nhandwritten, typewritten or printed text into machine-
    encoded\ntext. It is widely used as a form of data entry from some sort
    of\noriginal paper data source, whether documents, sales
    receipts,\nmail, or any number of printed records. It is a common
    method\nof digitizing printed texts so that they can be
    electronically\n\n",
    originalLanguage: "en",
    translatedText: "Οπτική αναγνώριση χαρακτήρων, συνήθως με τα αρχικά
    OCR, είναι η μηχανική ή ηλεκτρονική μετατροπή σαρωμένες εικόνες του
    χειρόγραφου, δακτυλογραφημένου ή τυπωμένου κειμένου στο μηχανή-
    κωδικοποιημένο κείμενο. It χρησιμοποιείται ευρέως ως μια μορφή της
    εισαγωγής δεδομένων από κάποιο είδος του αρχικού αρχείου προέλευσης
    δεδομένων χαρτί, είτε έγγραφα, παραλαβές πωλήσεων, ταχυδρομείο, ή
    οποιοδήποτε αριθμό τυπωμένο εγγραφές. It είναι μια κοινή μέθοδος της
    ψηφιοποίησης τυπωμένα κείμενα, έτσι ώστε να μπορούν να είναι
    ηλεκτρονικά\n\n"
}
```


Όπως φαίνεται είναι στη μορφή JSON που έχουμε ορίσει.

4.5 Εφαρμογή Android

Το Android app που κατασκευάσαμε ήταν σχετικά απλό από άποψη ρυθμίσεων, σε αντίθεση με το servlet που γράψαμε παραπάνω. Όπως θα φανεί στη διαδικασία δημιουργίας του project δεν ήταν απαραίτητος ο προσδιορισμός ενός πλήθους ρυθμίσεων όπως έγινε εξαιτίας των βιβλιοθηκών που χρησιμοποιήσαμε στο servlet.

Όπως αυτό δε σημαίνει πως δε συναντήσαμε απροσδόκητα εμπόδια. Το πρώτο από αυτά είχε να κάνει με τη διαδικασία αποστολής δεδομένων post με το πρωτόκολλο HTTP.

Ο λόγος για τον οποίο επιλέξαμε το πρωτόκολλο HTTP είναι η ευρύτατη διάδοσή του και η σχεδόν καθολική χρήση του από ένα πλήθος υπηρεσίες. Όμως όπως ανακαλύψαμε η λειτουργία αποστολής αρχείου με μία αίτηση HTTP post δεν υποστηρίζεται άμεσα από αυτό.

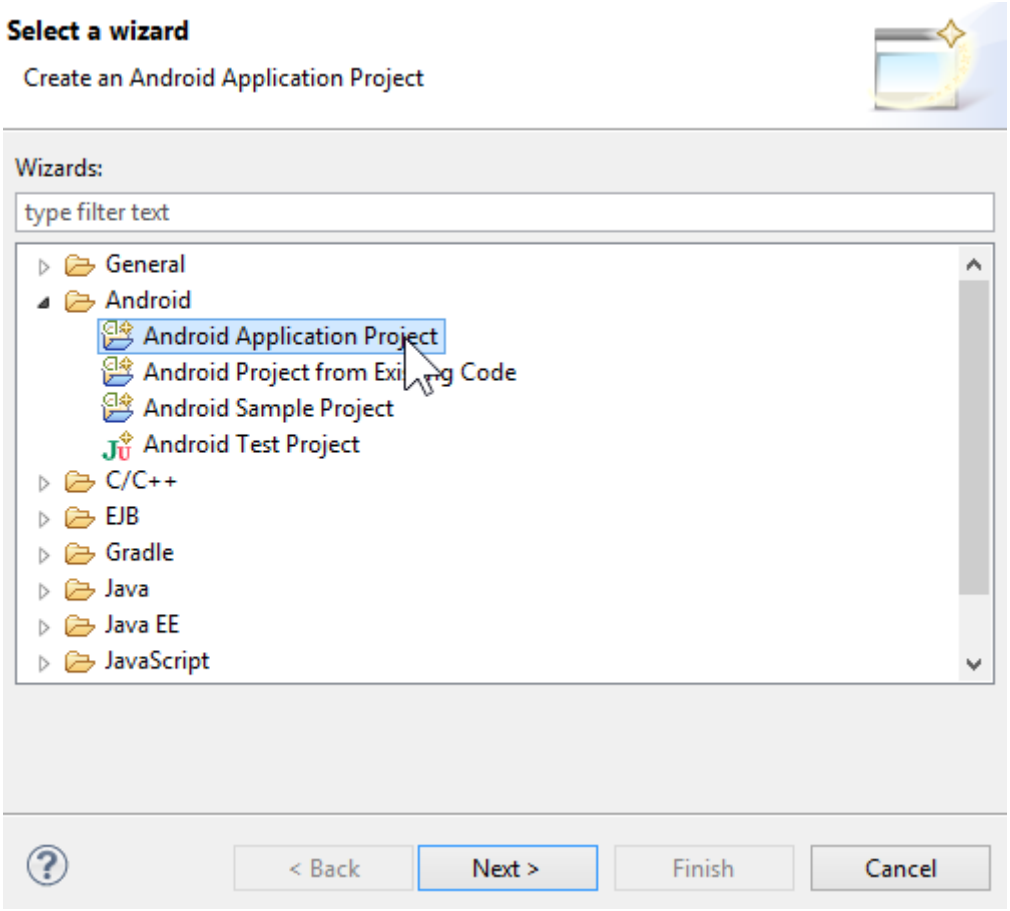
Το δεύτερο εμπόδιο έχει να κάνει με τον τρόπο που το Android απαιτεί να γίνονται οι συνδέσεις δικτύου. Συγκεκριμένα κάθε εφαρμογή έχει ένα κυρίως thread το οποίο χειρίζεται τα γραφικά και τις αιτήσεις του χρήστη. Για να υπάρχει μία καλύτερη εμπειρία χρήσης απαγορεύει να γίνονται μέσα από αυτό το thread λειτουργίες που σχετίζονται με το δίκτυο. Όπως τα εργαλεία που προσφέρει για την δημιουργία και το συντονισμό των επιπλέον threads είναι αρκετά βασικά και δύσχρηστα.

Επιπλέον η ανάγκη υποστήριξης πολλών εκδόσεων είναι πολλές φορές δύσκολη, καθώς δεν παρέχεται το κατάλληλο υλικό. Το documentation που υπάρχει στη σελίδα της Google εξηγεί σε μάκρος τις βασικές έννοιες αλλά δεν παρέχει λεπτομερή παραδείγματα κώδικα. Αντίθετα στην πλειοψηφία των περιπτώσεων παρέχονται κάποια σημαντικά σημεία του αντικειμένου που επεξεργάζεται και έπειτα ο χρήστης παραπέμπεται στο να κατεβάσει ολόκληρο το project του παραδείγματος, στο οποίο πολλές φορές δεν υπάρχουν ολοκληρωμένες εξηγήσεις για τα υπόλοιπα σημεία ενδιαφέροντος.

Ο ρυθμός με τον οποίο βγαίνουν νέες εκδόσεις είναι καταιγιστικός και εξαιτίας αυτού οποιοδήποτε βιβλίο ασχολείται με το θέμα περιέχει σίγουρα ανακρίβειες σχετικά με πράγματα που δεν ανήκουν στην πιο πρόσφατη έκδοση. Επίσης η υποστήριξη παλιότερων συσκευών γίνεται με ελαφρώς διαφορετικό κώδικα, ο οποίος κάποιες φορές δεν λειτουργεί στις νεότερες, καθιστώντας χρονοβόρα τη δοκιμή της εφαρμογής σε διαφορετικές εκδόσεις του λειτουργικού.

4.5.1 Δημιουργία project

Για τη δημιουργία του νέου project επιλέγουμε στο eclipse File →New → Project… και στο επόμενο παράθυρο επιλέγουμε το Android Application Project από την κατηγορία Android.



Στην επόμενη οθόνη χρειάζεται να δώσουμε κάποιες βασικές πληροφορίες για την εφαρμογή.

Το πεδίο Application name είναι το όνομα που θα εμφανίζεται να έχει η εφαρμογή μας. Αυτό είναι το όνομα που θα εμφανίζεται στον χρήστη και δεν χρησιμοποιείται πουθενά αλλού.

Το πεδίο Project name είναι το όνομα του project στο eclipse και πάλι δεν χρησιμοποιείται για προγραμματιστικούς σκοπούς.

Το πεδίο Package name είναι το όνομα του πακέτου μέσα στο οποίο θα βρίσκεται η εφαρμογή μας. Επιλέξαμε το gr.teiher.epp.ocrtranslator.app.

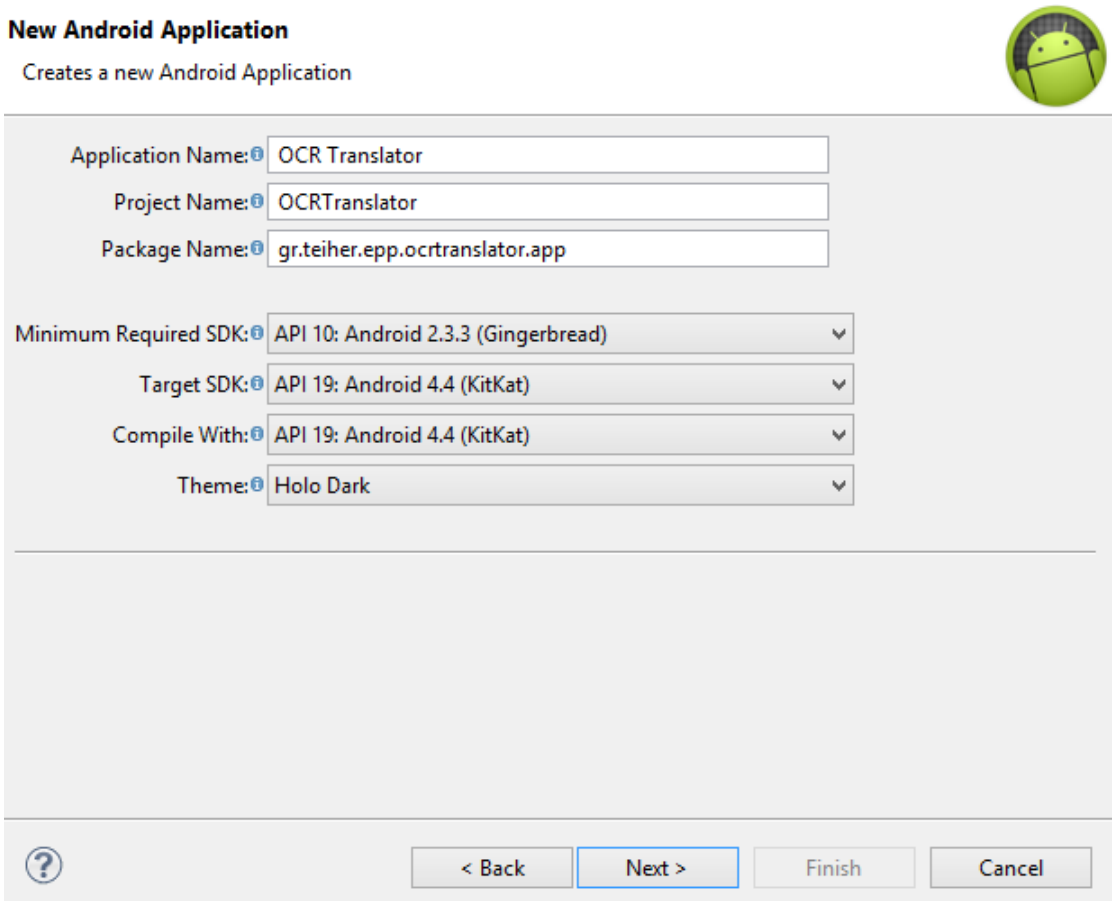
Παρακάτω επιλέγουμε τις εκδόσεις του Android που θα υποστηρίζει η εφαρμογή. Η minimum required version είναι η ελάχιστη έκδοση η οποία θα απαιτείται. Αυτή τη στιγμή η μικρότερη έκδοση που έχει αρκετά μεγάλο μερίδιο αγοράς ώστε να δικαιολογεί την υποστήριξή της είναι το Android 2.3.3. Οι εκδόσεις του Android έχουν τρία χαρακτηριστικά. Για παράδειγμα το Android 2.3.3 έχει κωδικό όνομα το Gingerbread και αναφέρεται σαν API level 10.

Η υποστήριξη παλιότερων εκδόσεων είναι μία αρκετά επίπονη και χρονοβόρα διαδικασία, και πολλές φορές απαιτεί την ύπαρξη διαφορετικών αρχείων της εφαρμογής (ή και ολόκληρων εφαρμογών) για διαφορετικές εκδόσεις. Κατά συνέπεια δεν έχει νόημα να επιδιώξουμε να υποστηρίξουμε όλες τις εκδόσεις αλλά μόνο αυτές που έχουν ακόμη ένα αξιόλογο μερίδιο αγοράς.

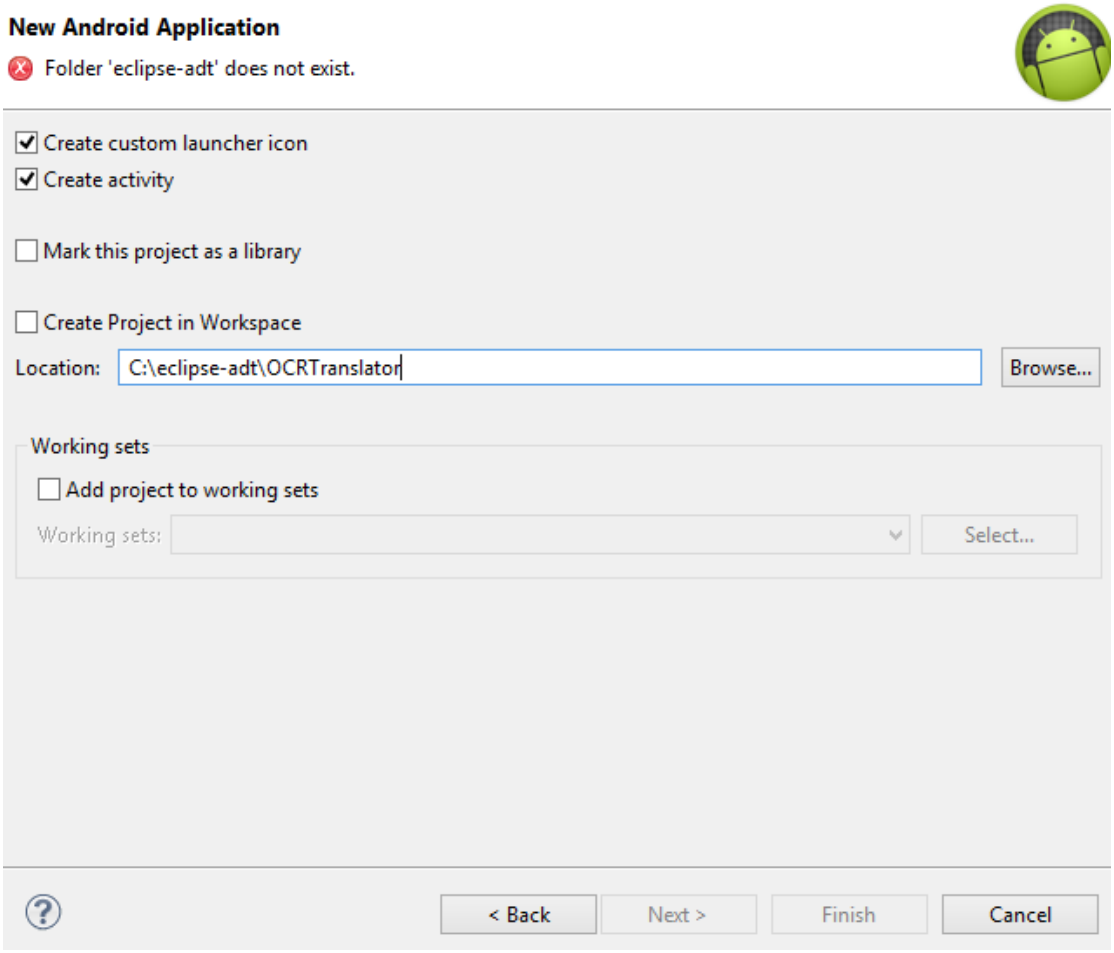
Το πεδίο Target SDK είναι η μεγαλύτερη έκδοση στην οποία έχει δοκιμαστεί η εφαρμογή. Εδώ επιλέγουμε την πιο πρόσφατη έκδοση του Android, που στην περίπτωσή μας είναι η 4.4 με API level 19.

Το πεδίο compile with έχει να κάνει με την έκδοση που θέλουμε να χρησιμοποιήσουμε για να γίνει compile η εφαρμογή. Εκτός και αν υπάρχει κάποια συγκεκριμένη ασυμβατότητα επιλέγουμε πάντοτε την πιο πρόσφατη έκδοση.

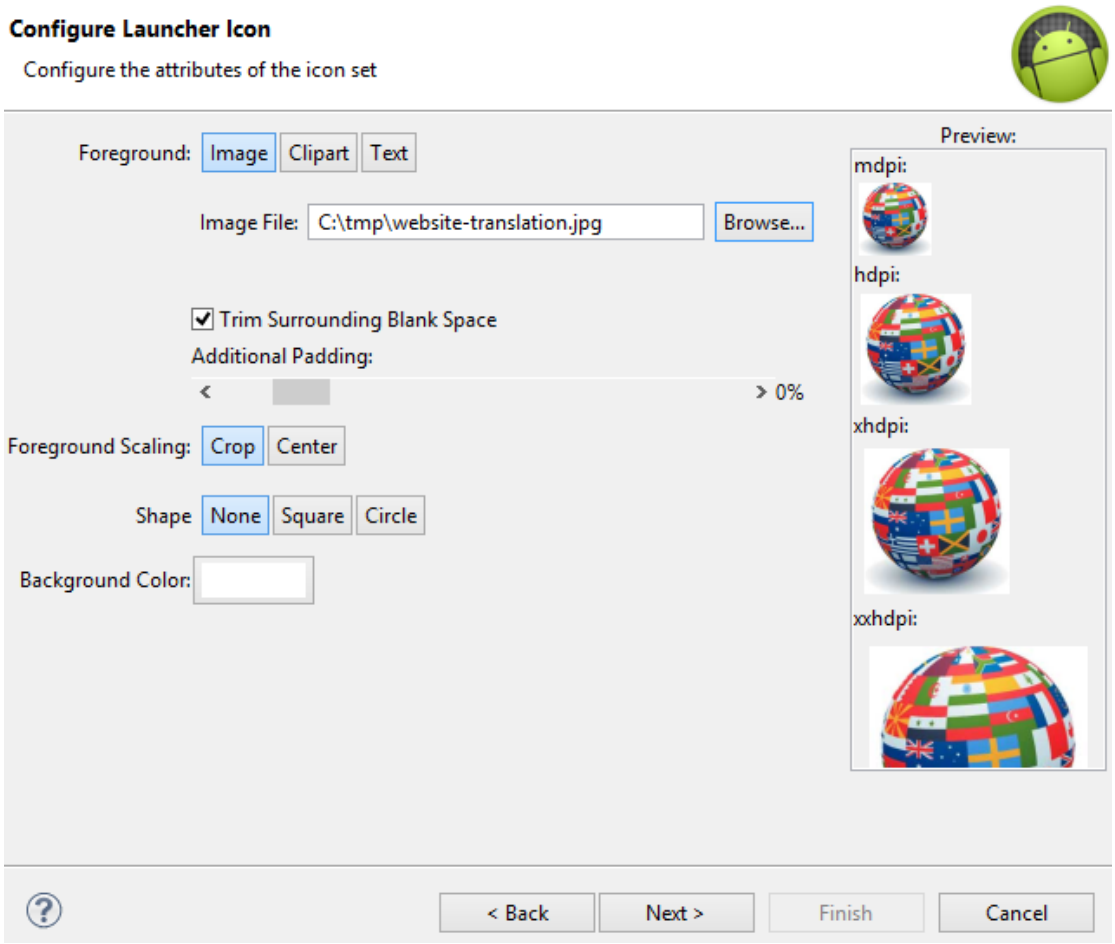
Τέλος το πεδίο theme έχει να κάνει με την εμφάνιση της εφαρμογής μας.



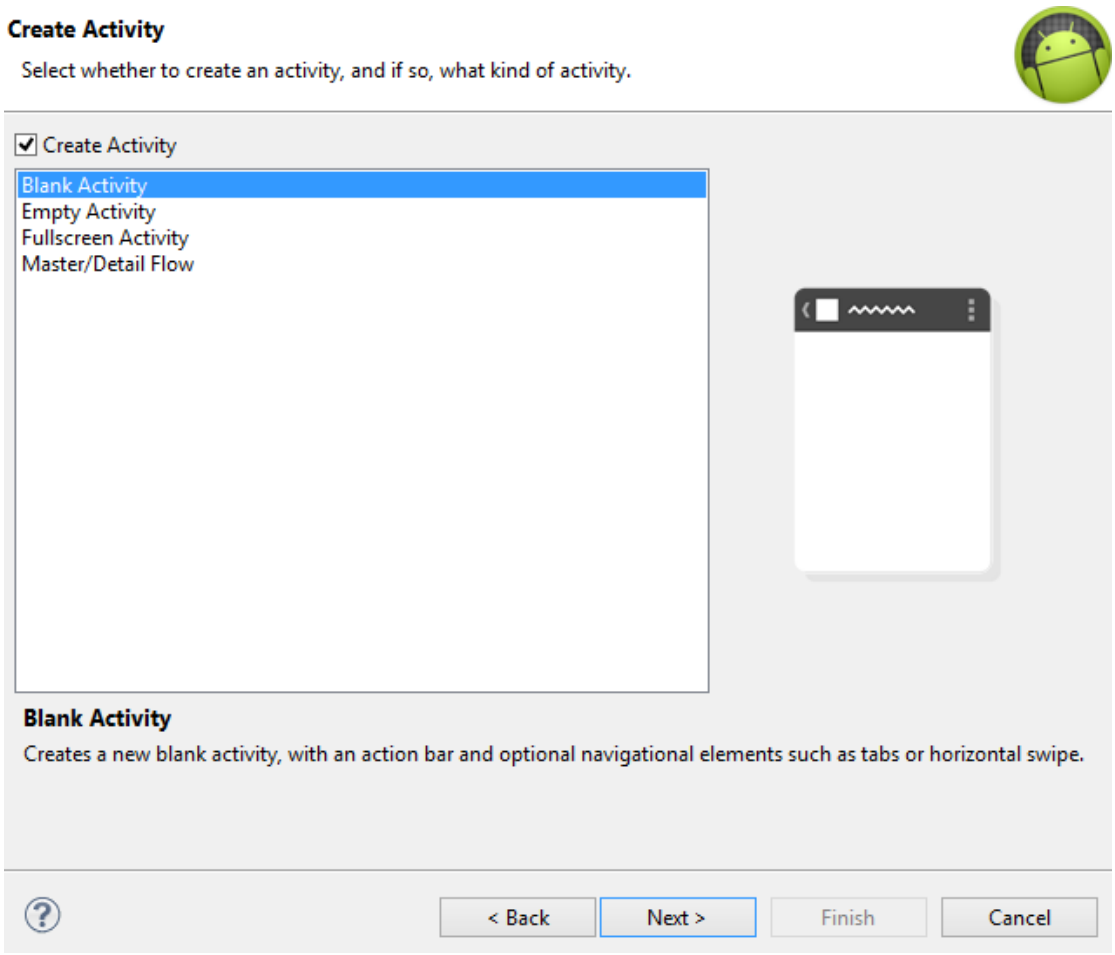
Στο επόμενο βήμα επιλέγουμε αν θέλουμε να δημιουργηθούν αυτόματα κάποια βασικά αρχεία. Δεν χρειάζεται να αλλάξουμε τίποτα εδώ.



Μετά μπορούμε να επιλέξουμε ένα εικονίδιο για την εφαρμογή μας. Αυτό αυτόματα μετατρέπεται σε όλες τις απαραίτητες διαστάσεις.



Έπειτα δημιουργούμε ένα καινούργιο activity. Αφού η εφαρμογή μας δεν θα έχει πολλές διαφορετικές οθόνες αρκεί να βάλουμε το blank activity.



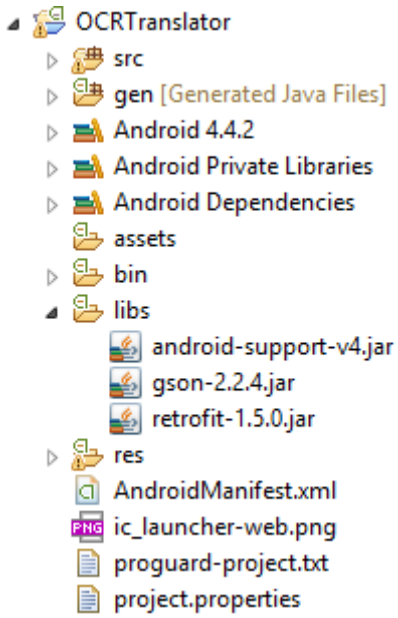
Για το activity που δημιουργήσαμε πρέπει να ορίσουμε τα βασικά στοιχεία του. Αυτά είναι το όνομά του και το όνομα του αρχείου που δημιουργεί. Αν και μπορούμε να τα αλλάξουμε αφήνουμε τις βασικές επιλογές.

Μετά από αυτό δημιουργείται το νέο project μας. Πρέπει να σημειωθεί ότι αυτόματα δημιουργείται και ένα νέο project, το οποίο ονομάζεται appcompat_v7_X. Αυτό είναι ορισμένο σαν project βιβλιοθήκης και περιέχει απαραίτητα στοιχεία ώστε να λειτουργούν όλα τα στοιχεία της εφαρμογής σε παλιότερες εκδόσεις. Για τη μεταφορά της εφαρμογής είναι απαραίτητα και τα δύο project.

Το X στο όνομα είναι ένας αύξον αριθμός και όχι ο αριθμός της έκδοσης. Η πρώτη εφαρμογή που θα δημιουργήσουμε στο eclipse θα έχει το appcompat_v7, για την δεύτερη το appcompat_v7_2, η Τρίτη θα συσχετιστεί αυτόματα με το appcompat_v7_3 κοκ.

Η εφαρμογή χρησιμοποιεί κάποιες βιβλιοθήκες με τη μορφή αρχείων jar. Αυτή τη φορά δεν χρειάζεται να κάνουμε πολύπλοκες ρυθμίσεις καθώς η εισαγωγή τους είναι σε μεγάλο βαθμό αυτοματοποιημένη. Συγκεκριμένα οποιοδήποτε αρχείο jar αντιγράψουμε μέσα στον φάκελο του project αυτόματα ορίζεται σαν απαραίτητο αρχείο βιβλιοθήκης και χρησιμοποιείται κατά τη μεταγλώτισση.

Τα αρχεία που χρειαστήκαμε είναι το retrofit-1.5.0.jar και το gson-2-2-4.jar. Το retrofit είναι μία βιβλιοθήκη που χρησιμοποιήσαμε για την επικοινωνία με τον server. Υποστηρίζει αυτόματα λήψη δεδομένων με μορφή JSON, όμως απαιτεί την ύπαρξη την κατάλληλης βιβλιοθήκης και συγκεκριμένα της gson.

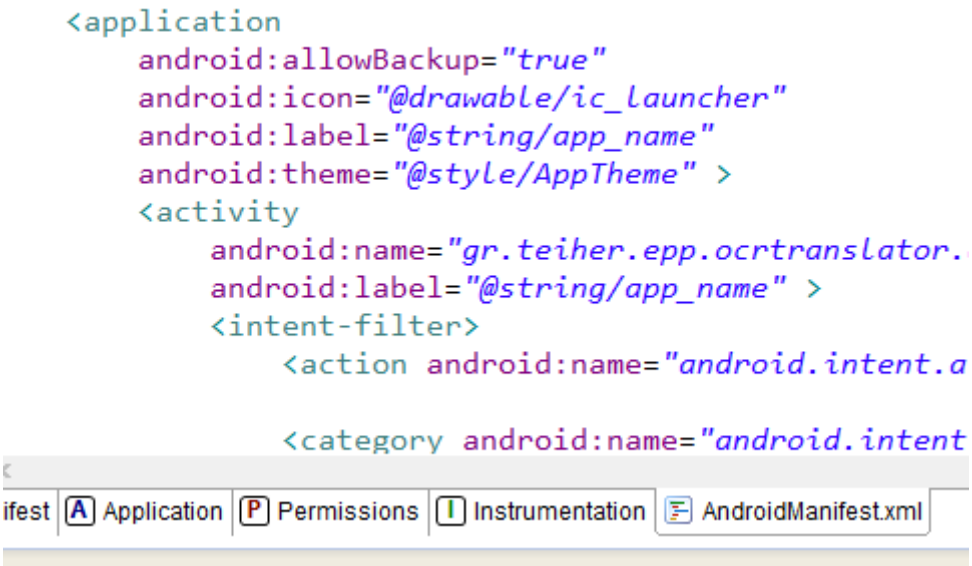


Μετά την αντιγραφή τους στον κατάλληλο φάκελο κάνουμε δεξί κλικ πάνω στο όνομα του project και επιλέγουμε Refresh. Τα εικονίδια των αρχείων αλλάζουν και εμφανίζουν το σύμβολο την βιβλιοθήκης στην κάτω δεξιά γωνία τους, σημάδι ότι έχουν αναγνωριστεί αυτόματα.

4.5.2 Επεξήγηση κώδικα

4.5.2.1 Αρχείο AndroidManifest.xml

Το αρχείο AndroidManifest.xml περιέχει γενικές πληροφορίες σχετικά με την εφαρμογή. Αυτό δημιουργείται αυτόματα κατά τη δημιουργία του project. Όταν το επιλέξουμε το eclipse μπορεί να εμφανίσει ένα γραφικό περιβάλλον για την επεξεργασία του, ενώ για την εμφάνιση του κώδικα XML πρέπει να επιλέξουμε το τελευταίο tab στο κάτω μέρος της σελίδας.



Σε αυτό το αρχείο πρέπει να δηλώσουμε τις δυνατότητες που χρειάζεται για να λειτουργήσει η εφαρμογή μας. Αυτές κατά την εγκατάσταση μίας εφαρμογής εμφανίζονται στον χρήστη ώστε να τις εγκρίνει ή να αρνηθεί την εγκατάσταση.

Μέσα στο αρχικό tag manifest προσθέτουμε τις παρακάτω γραμμές.

```
    <uses-feature android:name="android.hardware.camera"
                  android:required="true" />

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

Το τι κάνει η κάθε μία γίνεται εύκολα κατανοητό από το όνομά της. Η πρώτη γραμμή ενημερώνει ότι θα χρησιμοποιεί την κάμερα. Οι επόμενες 3 γραμμές για το internet, network state και wifi state ζητάνε άδεια για την πρόσβαση στο δίκτυο, καθώς θα είναι απαραίτητη η επικοινωνία με τον server. Τέλος η τελευταία γραμμή ζητάει άδεια για την εγγραφή δεδομένων στην κάρτα μνήμης.

Η φωτογραφία που θα βγάζουμε πρέπει να μπορεί να σωθεί πριν τη στείλουμε, κάτι το οποίο γίνεται μέσω αυτής της επιλογής.

4.5.2.2 Αρχείο string.xml

Κατά τη δημιουργία των αντικειμένων που περιέχει η εφαρμογή θεωρείται κακή πρακτική η απευθείας χρήση κειμένων, πχ για τα κουμπιά. Αντίθετα ενθαρρύνεται η μεταφορά του κειμένου στο αρχείο string.xml και η χρήση τους μέσω αυτού.

Το αρχείο φτιάχνεται αυτόματα και εμείς προσθέτουμε τις επιπλέον επιλογές. Το κείμενο που έχουμε εισάγει φαίνεται παρακάτω.

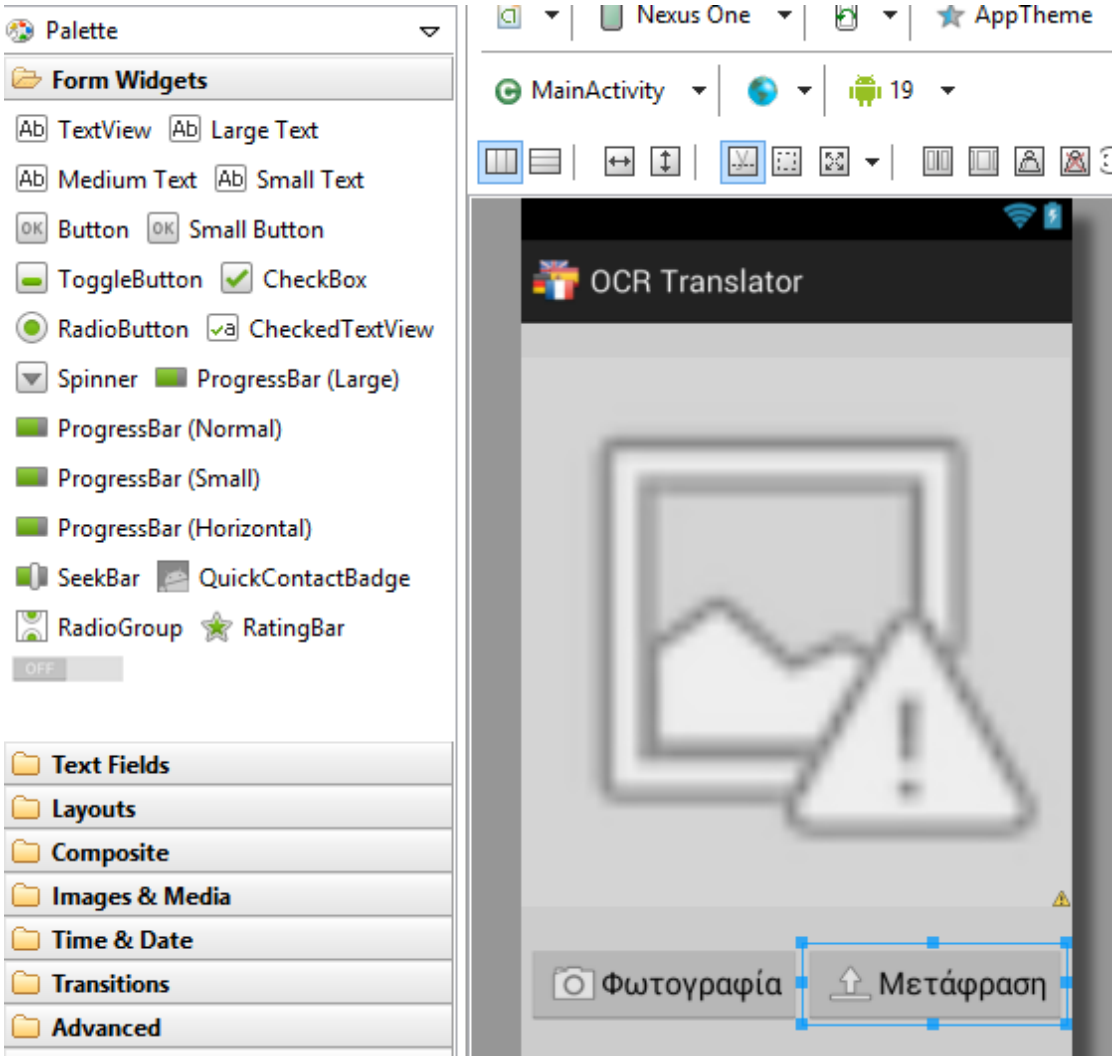
```
<string name="take_photo">Φωτογραφία</string>
<string name="translate">Μετάφραση</string>
<string name="server_address">Διεύθυνση server:</string>
<string name="server_port">Port:</string>
```

Αυτά τα string μπορούμε να τα χρησιμοποιήσουμε μελλοντικά μέσα στα αρχεία layout που θα χρησιμοποιήσουμε. Συγκεκριμένα η ιδιότητα με την οποία καθορίζουμε το κείμενο ενός κουμπιού είναι το android:text. Αντί να δώσουμε την εντολή

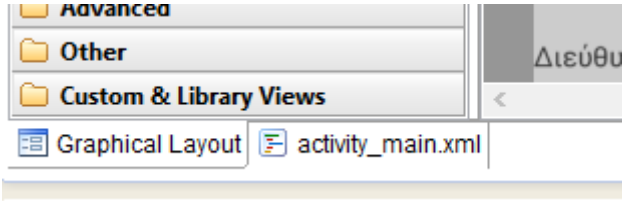
android:text="Φωτογραφία" γράφουμε android:text="@string/translate", όπου translate είναι το όνομα της εγγραφής που περιέχει τη λέξη φωτογραφία.

4.5.2.3 Αρχείο activity_main.xml

Πρόκειται για ένα αρχείο layout, δηλαδή ένα αρχείο που περιγράφει τα περιεχόμενα ενός activity και τις ιδιότητές τους. Ο editor του eclipse δίνει τη δυνατότητα να τοποθετήσουμε τα στοιχεία που θέλουμε στις κατάλληλες θέσεις γραφικά.



Όμως η χρήση του δεν είναι πάντα εύκολη. Κατ' αρχήν πολλές φορές δεν βγάζει σωστά αποτελέσματα, όπως παραπάνω που η κεντρική εικόνα φαίνεται πολύ μεγάλη. Επιπλέον δεν είναι πάντα εύκολο να τοποθετήσουμε τα αντικείμενα στις θέσεις που θέλουμε. Για αυτόν τον λόγο χρησιμοποιούμε κυρίως την μορφή xml, που την επιλέγουμε από το κατάλληλο tab στο κάτω μέρος.



Το βασικό στοιχείο που περιέχει όλα τα υπόλοιπα στοιχεία της εφαρμογής είναι ένα ScrollView, το οποίο το επιλέξαμε γιατί το ύψος της σελίδας μας δεν είναι σταθερό αλλά εξαρτάται από το μέγεθος της εικόνας που θα δοθεί και το μέγεθος του κειμένου που θα λάβουμε σαν απάντηση.

```
<ScrollView
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/scrollView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#cccccc" >
```

Μέσα σε αυτό έχουμε ένα LinearLayout με orientation vertical, ώστε να βάζει τα περιεχόμενά του το ένα κάτω από το άλλο.

```
<LinearLayout
      android:id="@+id/container"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:orientation="vertical"
      android:descendantFocusability="beforeDescendants"
      android:focusableInTouchMode="true"  >
```

Αυτό δεν έχει συγκεκριμένο ύψος αλλά μεγαλώνει αυτόματα ώστε να χωράνε τα περιεχόμενά του, με την οδηγία android:layout_height="wrap_content".

Μέσα του έχει την εικόνα. Αρχικά εκεί εμφανίζεται ένα βοηθητικό εικονίδιο αλλά όταν ο χρήστης πάρει την φωτογραφία εμφανίζεται αυτή στη θέση του.

```
<ImageView
    android:id="@+id/photoView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:src="@android:drawable/ic_menu_report_image"
    android:adjustViewBounds="true"
    android:layout_marginTop="20dp" />
```

Μετά την εικόνα έχουμε τα δύο κουμπιά για να πάρει ο χρήστης εικόνα και να την στείλει για μετάφραση. Επειδή αυτά θέλουμε να είναι δίπλα δίπλα τα βάζουμε μέσα σε ένα άλλο LinearLayout με horizontal orientation.

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="20dp"
    android:layout_marginTop="20dp"
    android:gravity="center_horizontal" >
```

Τα ίδια τα κουμπιά έχουν παρόμοιο κώδικα, με διαφορές στα id τους, την εικόνα που θα δείχνουν και το κείμενό τους.

```
<Button
    android:id="@+id/photoButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:drawableLeft="@android:drawable/ic_menu_camera"
    android:text="@string/take_photo" />

<Button
    android:id="@+id/translateButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:drawableLeft="@android:drawable/ic_menu_upload"
    android:text="@string/translate" />
```

Από κάτω ακολουθούν δύο LinearLayout που το καθένα έχει μέσα του ένα στοιχείο TextView που εμφανίζει κείμενο και ένα στοιχείο EditText για να μπορεί ο χρήστης να συμπληρώσει τη διεύθυνση και το port του server. Εδώ φαίνεται το ένα από αυτά.

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

    <TextView
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_weight="0.4"
        android:text="@string/server_address"
        android:textSize="16sp" />

    <EditText
        android:id="@+id/serverAddressText"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_weight="0.6" />
</LinearLayout>
```

Μετά από αυτά έχουμε ένα στοιχειο ProgressBar. Αυτό είναι μία μπάρα που αναλαμβάνει να δείξει στον χρήστη ότι χρειάζεται να περιμένει για να εκτελεστεί η προσδιορισμένη διαδικασία.

```
<ProgressBar
    android:id="@+id/loadingBar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:visibility="invisible" />
```

Αυτό αρχικά δεν φαίνεται, με την οδηγία android:visibility="invisible". Ο λόγος είναι πως θέλουμε να εμφανίζεται μόνο όταν ο χρήστης προσπαθήσει να επικοινωνήσει με τον server, κάτι που το κάνουμε προγραμματιστικά.

Τέλος υπάρχει ένα στοιχείο TextView που θα περιέχει την απάντηση που θα λάβει από τον server.

```
<TextView
    android:id="@+id/responseText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="14sp" />
```

Αυτό το στοιχείο δεν έχει κάποιο κείμενο οπότε αρχικά δε φαίνεται. Γιατί δεν πιάνει χώρο. Όταν ο χρήστης ζητήσει μία μετάφραση και δοθεί η απάντηση τότε το κάνουμε να τη δείχνει και μεγαλώνει αυτόματα.

4.5.2.4 Κλάση Config

Η κλάση Config δεν εκτελεί καμία λειτουργία, αλλά ορίζει κάποιες σταθερές που χρησιμοποιούνται στην εφαρμογή για να είναι εύκολο να τις αλλάξουμε.

```
public static final String DEFAULT_SERVER_ADDRESS =
"192.168.1.10";
public static final String DEFAULT_SERVER_PORT = "8080";

public static final String IMAGE_FILE_NAME = "IMAGE";
public static final String IMAGE_FILE_EXT = ".jpg";
```

Αυτές είναι η αρχική διεύθυνση και το port του server και το όνομα και η κατάληξη του αρχείου εικόνας. Αν θελήσουμε να βάλουμε διαφορετικές τιμές αρκεί να τις αλλάξουμε σε αυτό το σημείο αντί να τις ψάχνουμε στον κώδικα.

4.5.2.5 Κλάση OcrData

Η κλάση OcrData είναι μία βοηθητική κλάση που περιγράφει ένα αντικείμενο με παρόμοιες μεταβλητές με την απάντηση JSON που στέλνει ο server. Το ουσιαστικό τμήμα του κώδικα είναι οι δηλώσεις των μεταβλητών.

```
    @SerializedName("originalText")
    private String originalText;

    @SerializedName("originalLanguage")
    private String originalLanguage;

    @SerializedName("translatedText")
    private String translatedText;
```

Τα annotations που υπάρχουν πάνω από την κάθε μία δίνουν οδηγίες για το όνομα που έχει η αντίστοιχη μεταβλητή στο αρχείο JSON. Η απόδοση τιμών από το JSON στο αντικείμενο γίνεται αυτόματα με τη βιβλιοθήκη retrofit.

4.5.2.6 Interface OcrService

Το interface OcrService χρησιμοποιείται από τη βιβλιοθήκη Retrofit. Περιέχει τη δήλωση μόνο μίας μεθόδου:

```
@Multipart
@POST("/OCR_Translator/ProcessImage")
void sendImage(@Part("imageFile") TypedFile photo,
Callback<OcrData> callback);
```

Με το retrofit, όταν θέλουμε να κάνουμε μία αίτηση δεν δηλώνουμε τις πληροφορίες κάθε φορά που την κάνουμε. Αντίθετα δηλώνουμε κάποιες μεθόδους σε ένα interface με τέτοιο τρόπο ώστε να περιγράφουν τη διεύθυνση, τα δεδομένα που στέλνονται και το τι επιστρέφει.

Εδώ λέμε ότι τα δεδομένα θέλουμε να σταλούν σαν μία αίτηση post / multipart στην κατάλληλη διεύθυνση. Η πρώτη παράμετρος λέει ότι θα αποστέλλεται ένα αρχείο, το οποίο θα δίνεται σαν τιμή στη μεταβλητή photo.

Η δεύτερη παράμετρος δίνει ένα Callback, το οποίο θα καλείται όταν έρθει η απάντηση. Επειδή ο τύπος του είναι το OcrData μπορεί να καταλάβει ότι θα λάβει μία απάντηση με τη μορφή JSON, τα στοιχεία της οποίας θα τα βάλει στα αντίστοιχα πεδία ενός αντικειμένου OcrData.

4.5.2.7 Κλάση MainActivity

Η κλάση MainActivity περιέχει τις λειτουργίες του κυρίως activity (οθόνης) της εφαρμογής μας.

Ξεκινάει να τρέχει από την μέθοδο onCreate και αρχικά δηλώνει ότι θα χρησιμοποιήσει το αρχείο layout που δηλώσαμε παραπάνω.

```
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
```

Μετά βάζει σε εσωτερικές μεταβλητές τα στοιχεία της φόρμας που θέλουμε να χρησιμοποιήσουμε προγραμματιστικά.

```
            photoView = (ImageView)
findViewById(R.id.photoView);
            loadingBar = (ProgressBar)
findViewById(R.id.loadingBar);
            responseText = (TextView)
findViewById(R.id.responseText);
```

Ειδικά στα πεδία κειμένου βάζει σαν αρχικές τιμές τις τιμές που παίρνει από το Config.

```
serverAddressText = (EditText)
findViewById(R.id.serverAddressText);
      serverAddressText.setText(Config.DEFAULT_SERVER_ADDRESS);

            serverPortText = (EditText)
findViewById(R.id.serverPortText);
            serverPortText.setText(Config.DEFAULT_SERVER_PORT);
```

Μετά βάζει στα δύο κουμπιά τους listeners για το τι θα κάνουν όταν τα πατάει ο χρήστης.

Το πρώτο κουμπί αφήνει τον χρήστη να πάρει μία φωτογραφία. Πριν από αυτό πρέπει να φτιάξουμε το αρχείο στο οποίο θα αποθηκευτεί.

```
photoFile = File.createTempFile(Config.IMAGE_FILE_NAME,
Config.IMAGE_FILE_EXT, getExternalCacheDir());
photoFilePath = "file:" + photoFile.getAbsolutePath();
```

Το όνομα του αρχείου βγαίνει από τις τιμές που έχουμε δώσει στο Config και του λέμε να το αποθηκεύσει στο φάκελο προσωρινής μνήμης. Μετά παίρνουμε και το πλήρες path του για να το χρησιμοποιήσουμε παρακάτω.

Έπειτα κατασκευάζουμε το κατάλληλο Intent με το οποίο λέμε ότι θέλουμε να τραβήξουμε μία φωτογραφία, η οποία θα αποθηκευτεί στο αρχείο που δώσαμε.

```
Intent takePictureIntent = new
Intent(MediaStore.ACTION_IMAGE_CAPTURE);
takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT,
Uri.fromFile(photoFile));
startActivityForResult(takePictureIntent,
REQUEST_IMAGE_CAPTURE);
```

Με το startActivityForResult ο έλεγχος φεύγει από το πρόγραμμά μας.

Όταν ο χρήστης πάρει τη φωτογραφία θα καλεστεί αυτόματα η μέθοδος onActivityResult.

```
Bitmap imageBitmap = resizeImage();

photoView.setImageBitmap(imageBitmap);
```

Αυτή αλλάζει το μέγεθος της εικόνας που θα δείξει και έπειτα το βάζει στο αντίστοιχο ImageView ώστε να εμφανιστεί.

Το δεύτερο κουμπί αναλαμβάνει να στείλει την εικόνα στον server.

```
responseText.setText("");
loadingBar.setVisibility(View.VISIBLE);
sendImage();
```

Αυτό σβήνει το κείμενο από τυχών προηγούμενη απάντηση, εμφανίζει το progress bar για να μπορεί να καταλάβει ο χρήστης ότι η διαδικασία εξελίσσεται και καλεί την κατάλληλη συνάρτηση sendImage.

Το πρώτο πράγμα που γίνεται είναι να δημιουργηθεί η διεύθυνση του server από τις τιμές των πεδίων κειμένου.

```
String serverAddress =
          "http://"
          + serverAddressText.getText()
          + ":"
          + serverPortText.getText();
```

Μετά κατασκευάζουμε το απαραίτητο αντικείμενο στο Retrofit με αυτή τη διεύθυνση.

```
restAdapter = new RestAdapter.Builder()
            .setEndpoint(serverAddress)
            .build();
```

Έπειτα φτιάχνουμε το service που δηλώσαμε παραπάνω για την αποστολή των δεδομένων.

```
OcrService ocrService = restAdapter.create(OcrService.class);
TypedFile f = new TypedFile("image/jpeg", photoFile);
ocrService.sendImage(f, new Callback<OcrData>() {…
```

Σε αυτό δίνουμε σαν παραμέτρους το αρχείο που αποθηκεύσαμε την εικόνα και ένα callback που θα καλεστεί όταν έρθει η απάντηση.

Από το callback μας ενδιαφέρει το τι θα γίνεται στη μέθοδο success, δηλαδή όταν λάβει την απάντηση.

Αρχικά κρύβουμε το progress bar αφού σταμάτησε να φορτώνει.

```
loadingBar.setVisibility(View.INVISIBLE);
```

Το κείμενο της απάντησης θα το βάλουμε σε ένα TextView. Αυτό δεν μπορεί να δείξει μορφοποιημένο κείμενο, εκτός και αν είναι σε μορφή HTML. Για αυτό το λόγο αντικαθιστούμε τόσο από το αρχικό κείμενο όσο και από την απάντηση το \n με το
, με το οποίο αλλάζει γραμμή στην HTML.

```
data.setOriginalText(data.getOriginalText().replaceAll("\n",
"<br>"));
data.setTranslatedText(data.getTranslatedText().replaceAll("\n
", "<br>"));
```

Έπειτα δημιουργούμε ένα String στο οποίο συνθέτουμε το κείμενο HTML που θέλουμε να δείξουμε.

```
String res = "<h1>Original text</h1><p>" +
data.getOriginalText() + "</p>";
res += "<h1>Language: " + data.getOriginalLanguage()
+"</h1>";
res += "<h1>Translation</h1><p>" + data.getTranslatedText() +
"</p>";
```

Συγκεκριμένα βάζουμε σαν τίτλο το Original text και μετά μέσα σε μία παράγραφο το κείμενο που αναγνώρισε το Tessetact. Έπειτα βάζουμε σαν τίτλο τη γλώσσα που ανιχνεύτηκε για αυτό το κείμενο και ακολουθεί το μεταφρασμένο κείμενο.

Τέλος βάζουμε αυτό το κείμενο στο TextView της απάντησης.

```
responseText.setText(Html.fromHtml(res));
```

Αφού δίνουμε κείμενο HTML πρέπει να χρησιμοποιήσουμε την ειδική συνάρτηση fromHtml.

5 Οδηγίες εκτέλεσης εφαρμογής

5.1 Γενικά

Σε αυτό το κεφάλαιο θα περιγράψουμε αναλυτικά τα βήματα που πρέπει να γίνουν ώστε να τρέξει η εφαρμογή με τη χρήση των παραδοτέων που παρέχουμε.

Επειδή θα έχουμε δύο eclipse που θα τρέχουν ταυτόχρονα, όπου υπάρχει πιθανότητα σύγχυσης θα τα προσδιορίζουμε με τους όρους eclipse-JEE και eclipse-ADT

5.2 Εγκατάσταση εργαλείων

5.2.1 Εγκατάσταση JDK

Αρχικά χρειάζεται να κατεβάσουμε και να εγκαταστήσουμε το JDK για να μπορούμε να τρέχουμε και να κάνουμε compile προγράμματα Java. Αυτό γίνεται όπως στην ενότητα 3.3. Πρέπει να επιλέξουμε την τελευταία έκδοση της Java 7. Αν και μελλοντικές εκδόσεις πιθανότατα θα δουλεύουν δεν έχουν δοκιμαστεί στην εφαρμογή μας.

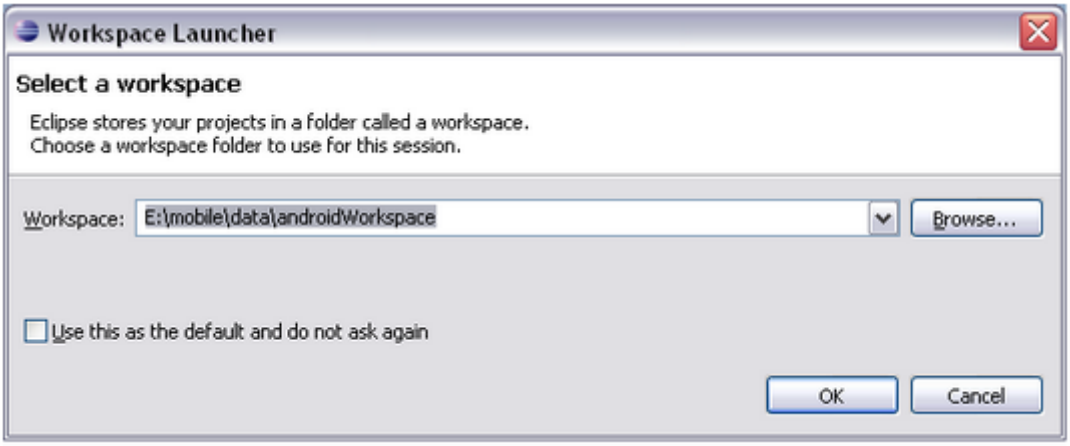
Επίσης πρέπει να επιλεγεί οπωσδήποτε έκδοση 32 bit. Αν υπάρχει εγκαταστημένη στο μηχάνημα και μία έκδοση της Java 64 bit μπορεί να δημιουργήσει προβλήματα στην πορεία.

5.2.2 Εγκατάσταση Tomcat

Εξαιτίας των ρυθμίσεων που πρέπει να γίνουν στον Tomcat, περιλαμβάνουμε τον server που χρησιμοποιήσαμε στα παραδοτέα. Για τη χρήση του αρκεί να αποσυμπιεστεί σε έναν φάκελο, χωρίς να απαιτούνται ρυθμίσεις σε αυτό το βήμα.

5.2.3 Εγκατάσταση eclipse για το web app

Αρχικά χρειάζεται να κατεβάσουμε το eclipse (στο οποίο θα αναφερόμαστε με τον όρο eclipse-JEE) όπως περιγράφεται στην ενότητα 3.6 και να το αποσυμπιέσουμε σε έναν φάκελο της επιλογής μας. Πριν το τρέξουμε δημιουργούμε έναν φάκελο για το workspace του, καθώς το ζητάει την πρώτη φορά που τρέχει.



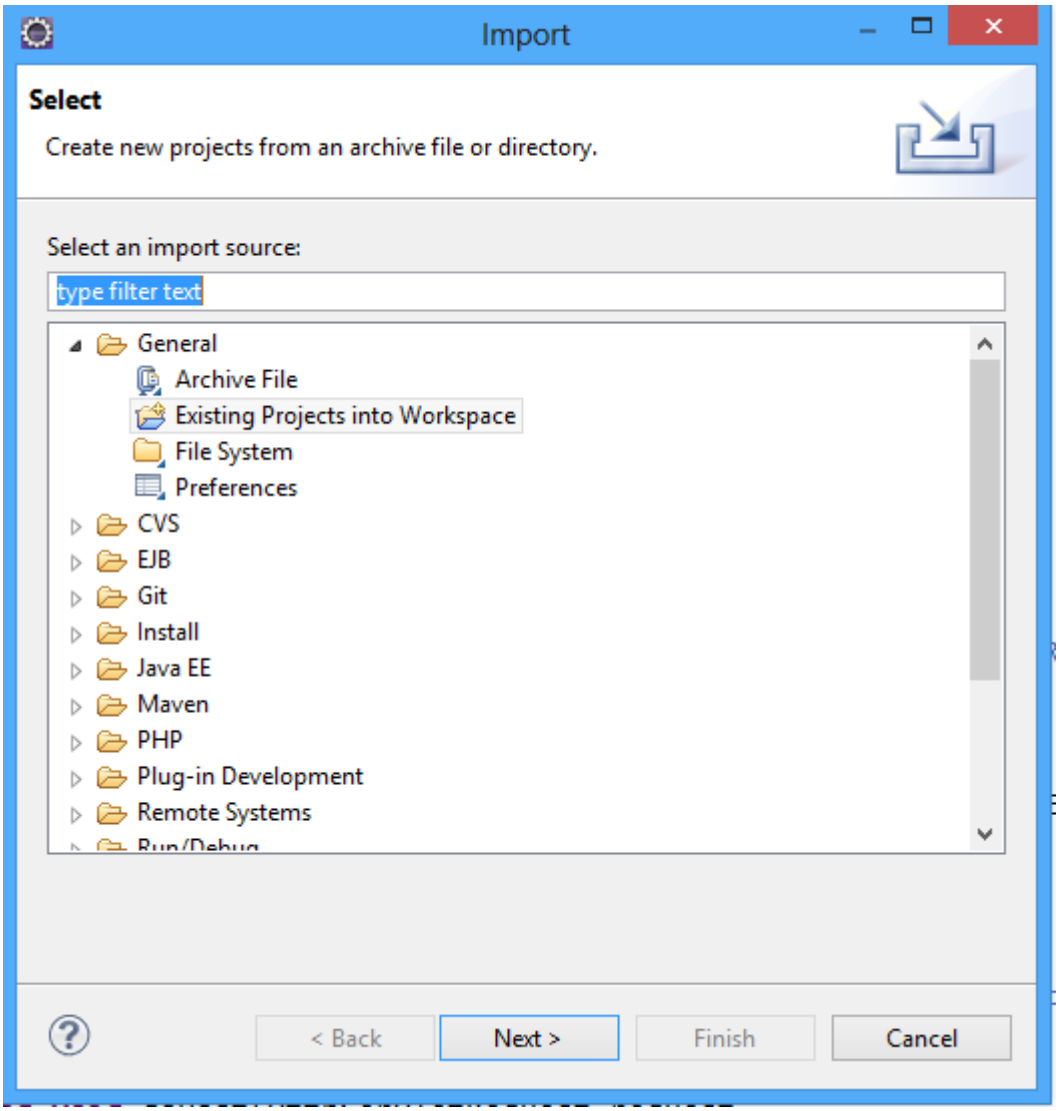
Για να μην κάνει συνέχεια αυτή την ερώτηση επιλέγουμε το Use this as the default and do not ask again.

5.2.4 Ρύθμιση server

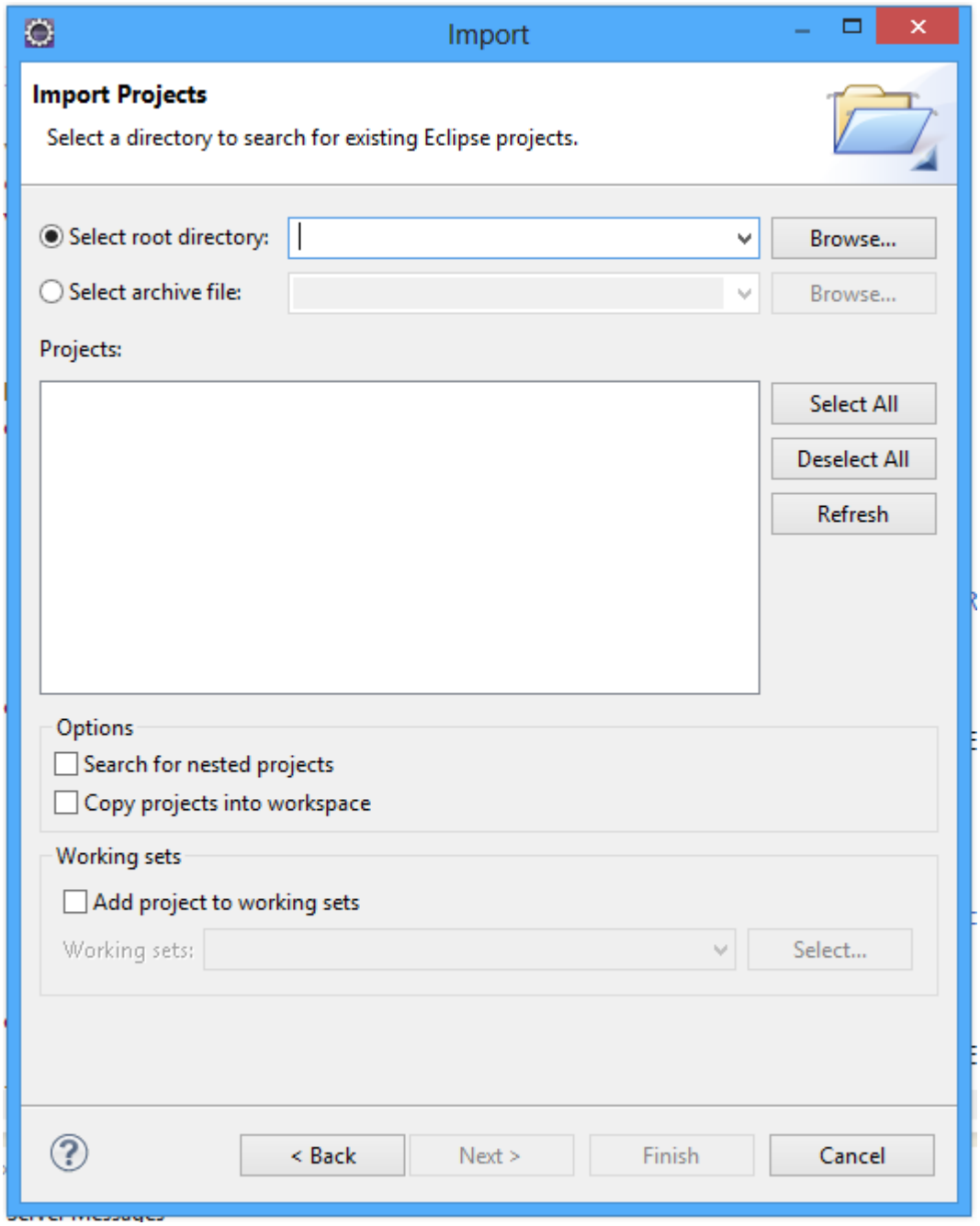
Μετά είναι απαραίτητο να ορίσουμε τον server μέσα από το eclipse όπως περιγράφεται στην ενότητα 4.4.1.

5.2.5 Εισαγωγή web project

Στο επόμενο βήμα εισάγουμε το project μας στο eclipse. Αρχικά αποσυμπιέζουμε τα περιεχόμενα του αρχείου OCR Translator - Server.zip στον φάκελο που δηλώσαμε προηγουμένως σαν workspace για το eclipse-JEE. Μετά, μέσα από το eclipse επιλέγουμε File → Import… και στο παράθυρο που εμφανίζεται επιλέγουμε General → Existing Projects into Workspace.



Στο επόμενο παράθυρο πατάμε το πρώτο κουμπί Browse δίπλα από το Select root directory και πατάμε το Finish.

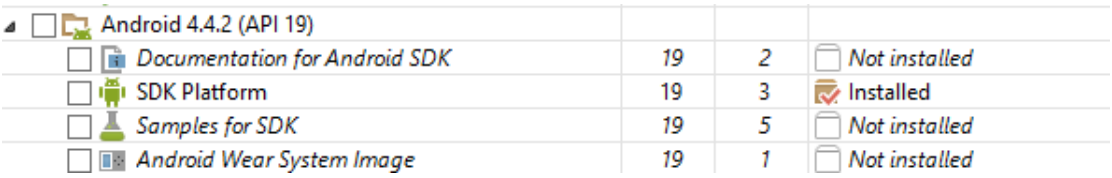


Αυτό θα πρέπει να ανοίξει το project.

5.2.6 Εγκατάσταση ADT

Έπειτα κατεβάζουμε το ADT και το αποσυμπιέζουμε μέσα σε έναν φάκελο. Καθώς και αυτό περιέχει μία έκδοση του eclipse μέσα στον ομώνυμο φάκελο (στο οποίο θα αναφερόμαστε με το όνομα eclipse-ADT παρακάτω) πριν το τρέξουμε πρέπει να δημιουργήσουμε έναν επιπλέον φάκελο για το workspace του, διαφορετικό από το workspace του eclipse-JEE.

Ακολουθώντας τις οδηγίες της ενότητας 3.4 κατεβάζουμε τα επιπλέον αρχεία που χρειάζονται από το SDK Manager. Πρέπει να υπάρχουν οπωσδήποτε τα αρχεία για τις εκδόσεις API19 και API10, που χρησιμοποιεί ρητά η εφαρμογή μας.



Αυτό αναφέρεται στην επιλογή SDK platform, όπως φαίνεται στην παραπάνω εικόνα για την έκδοση 19. Αν δεν γράφει στη δεξιά στήλη το installed, πρέπει να το επιλέξουμε και να πατήσουμε το κουμπί Install.

Η ίδια διαδικασία πρέπει να γίνει και για την ενότητα Android 2.3.3 (API 10).

5.2.7 Εισαγωγή project για το mobile app

Για να εισάγουμε το project στο eclipse είναι απαραίτητο να αποσυμπιέσουμε τα περιεχόμενα του αρχείου OCRTranslator App.zip στον φάκελο που ορίσαμε σαν workspace του eclipse-ADT. Έπειτα εισάγουμε τα project όπως ήδη δείξαμε στην ενότητα 5.2.5.

Αυτή η διαδικασία πρέπει να γίνει δύο φορές. Την πρώτη πρέπει να διαλέξουμε τον φάκελο του project appcompat_7_3 που χρησιμοποιεί σαν βιβλιοθήκη η εφαρμογή μας και τη δεύτερη τον ίδιο το φάκελο της εφαρμογής.

Αν αυτή η διαδικασία δεν γίνει με αυτή τη σειρά μπορεί να παρουσιαστούν προβλήματα στην πορεία.

5.3 Εκτέλεση εφαρμογής

Για την εκτέλεση της εφαρμογής χρειάζεται προφανώς μία συσκευή τηλεφώνου. Τόσο ο υπολογιστής που τρέχει τον server όσο και η εφαρμογή πρέπει να βρίσκονται συνδεδεμένα στο ίδιο (ασύρματο) δίκτυο ώστε να μπορούν να επικοινωνήσουν.

5.3.1 Εκτέλεση server

Αρχικά χρειάζεται να ανοίξουμε το eclipse-JEE και να τρέξουμε το Project κάνοντας δεξί κλικ πάνω του και επιλέγοντας Run on server. Αυτό θα ανοίξει και μία σελίδα του browser που θα βγάλει ένα μήνυμα λάθους, κάτι απόλυτα φυσιολογικό γιατί δεν έχουμε αρχική σελίδα (αφού μας ενδιαφέρει να μιλάει μόνο με το κινητό).

Παρόλα αυτά από τη διεύθυνση που θα έχει αυτή η σελίδα κρατάμε το port που δείχνει, συνήθως το 8080.

Μετά πρέπει να βρούμε τη διεύθυνση IP του μηχανήματος που τρέχει τον server, κάτι που γίνεται μέσα από τις ρυθμίσεις των windows.

5.3.2 Εκτέλεση εφαρμογής

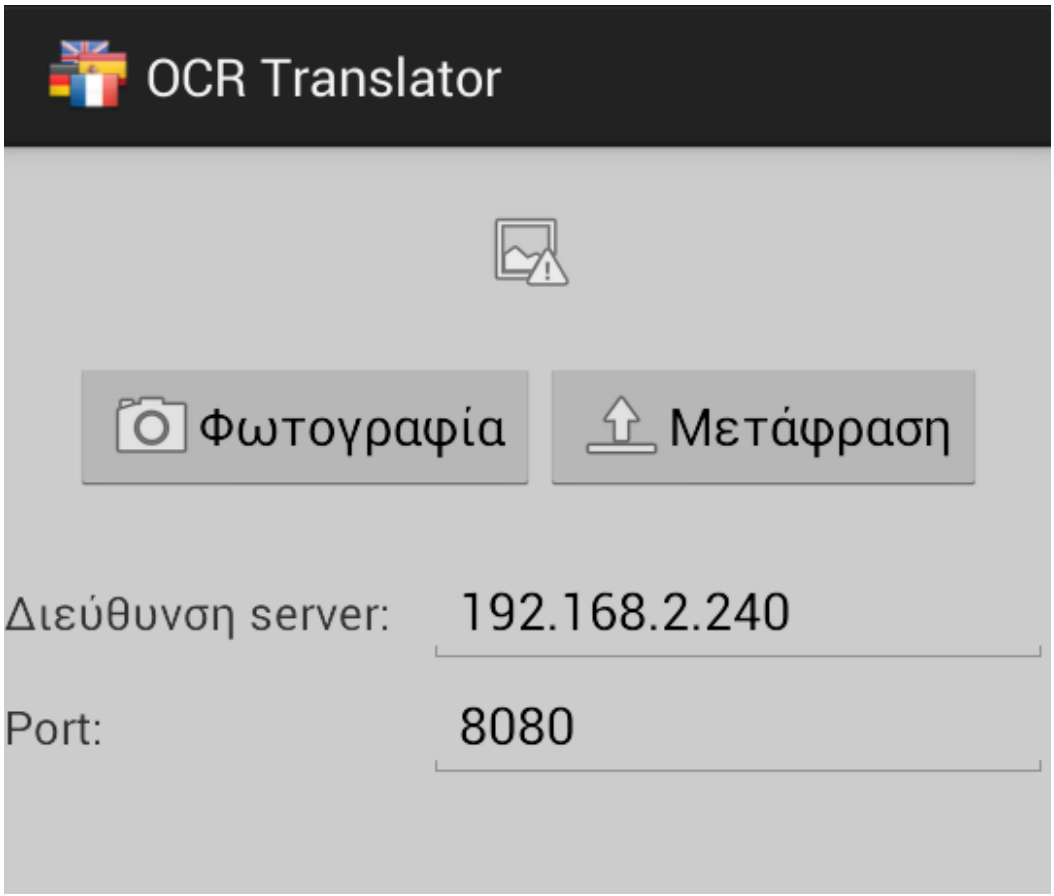
Για την εκτέλεση της εφαρμογής συνδέουμε το κινητό στον υπολογιστή με ένα καλώδιο USB. Για τα περισσότερα κινητά πρέπει να εγκαταστήσουμε προηγουμένως το κατάλληλο πρόγραμμα της κατασκευάστριας εταιρείας ώστε να μπορεί να το δει το λειτουργικό σύστημα.

Έπειτα τρέχουμε το eclipse-ADT. Σε αυτό κάνουμε δεξί κλικ επάνω στο project και διαλέγουμε το Run as → Android application.

Αφού τρέξει η εφαρμογή στο κινητό πρέπει να συμπληρώσουμε στα κατάλληλα πεδία τη διεύθυνση IP του server και το port που βρήκαμε στο προηγούμενο βήμα.

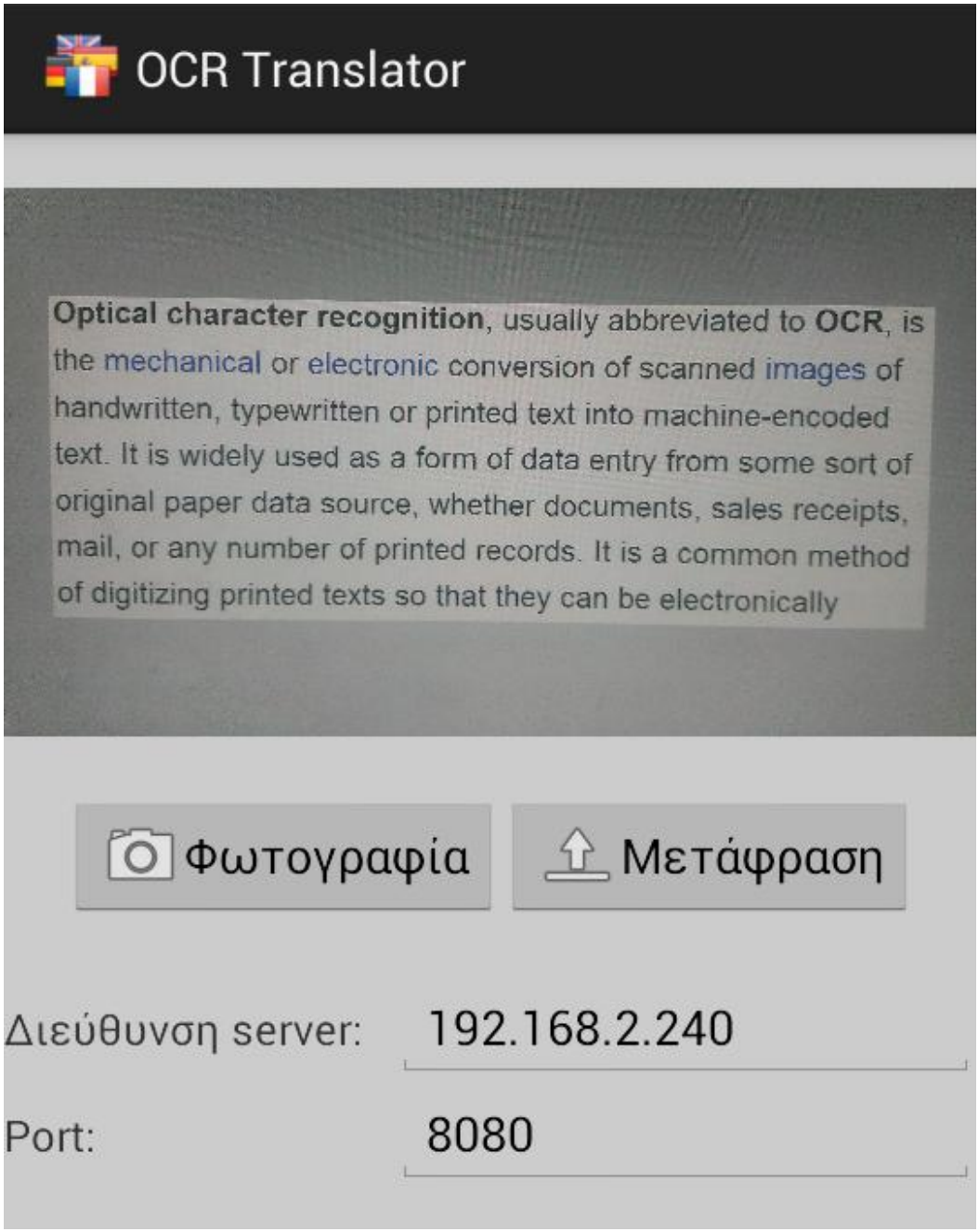
5.3.3 Οδηγίες χρήσης

Όταν ο χρήστης τρέχει την εφαρμογή στο κινητό εμφανίζεται η παρακάτω οθόνη:

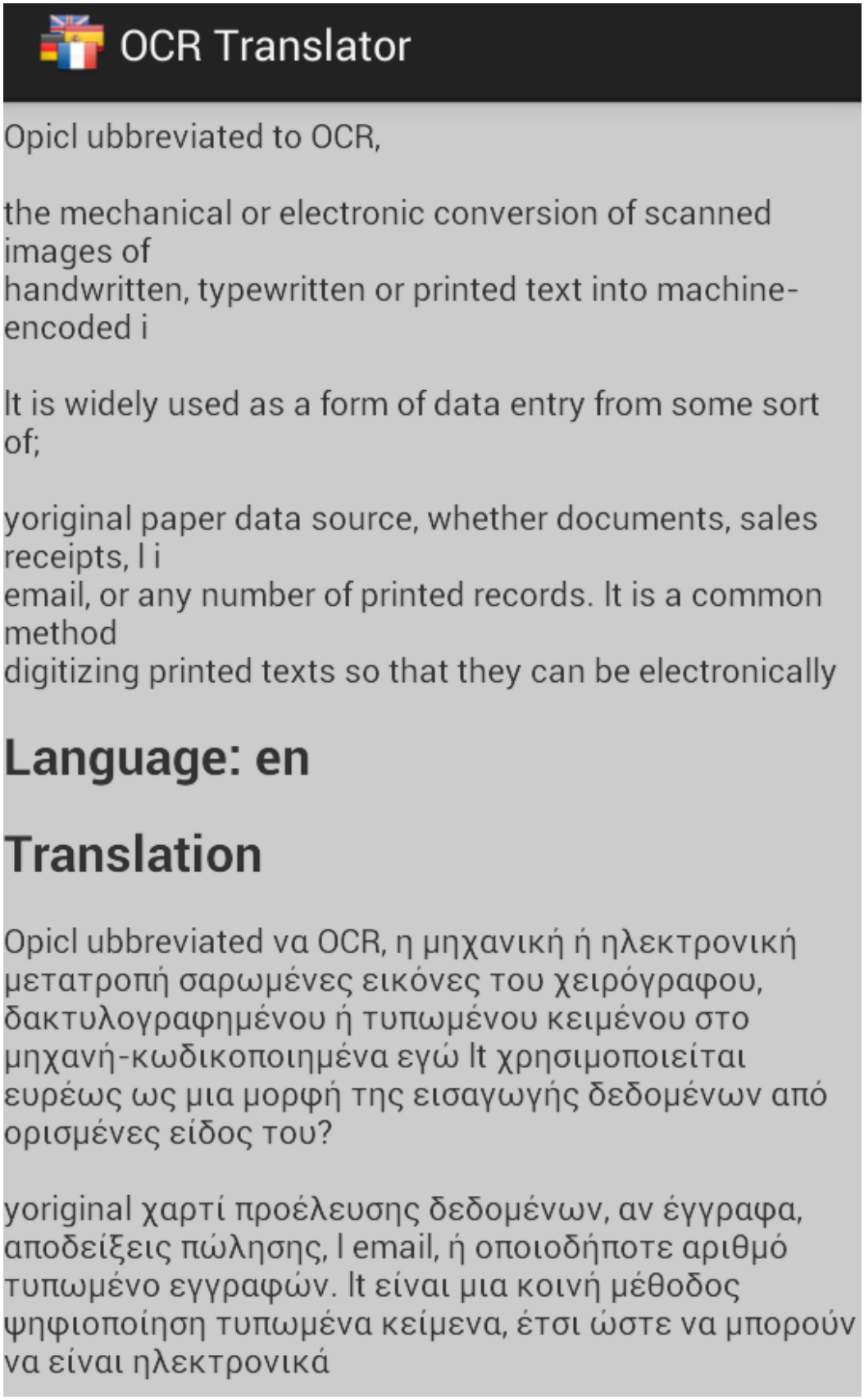


Από εκεί αρχικά επιλέγει τη διεύθυνση στην οποία βρίσκεται ο server και αν χρειάζεται το port του.

Έπειτα πατώντας το κουμπί φωτογραφία καλείται να φωτογραφίσει το κείμενο που θέλει να μεταφράσει. Μετά τη λήψη εμφανίζεται μία μικρογραφία της φωτογραφίας για επαλήθευση.

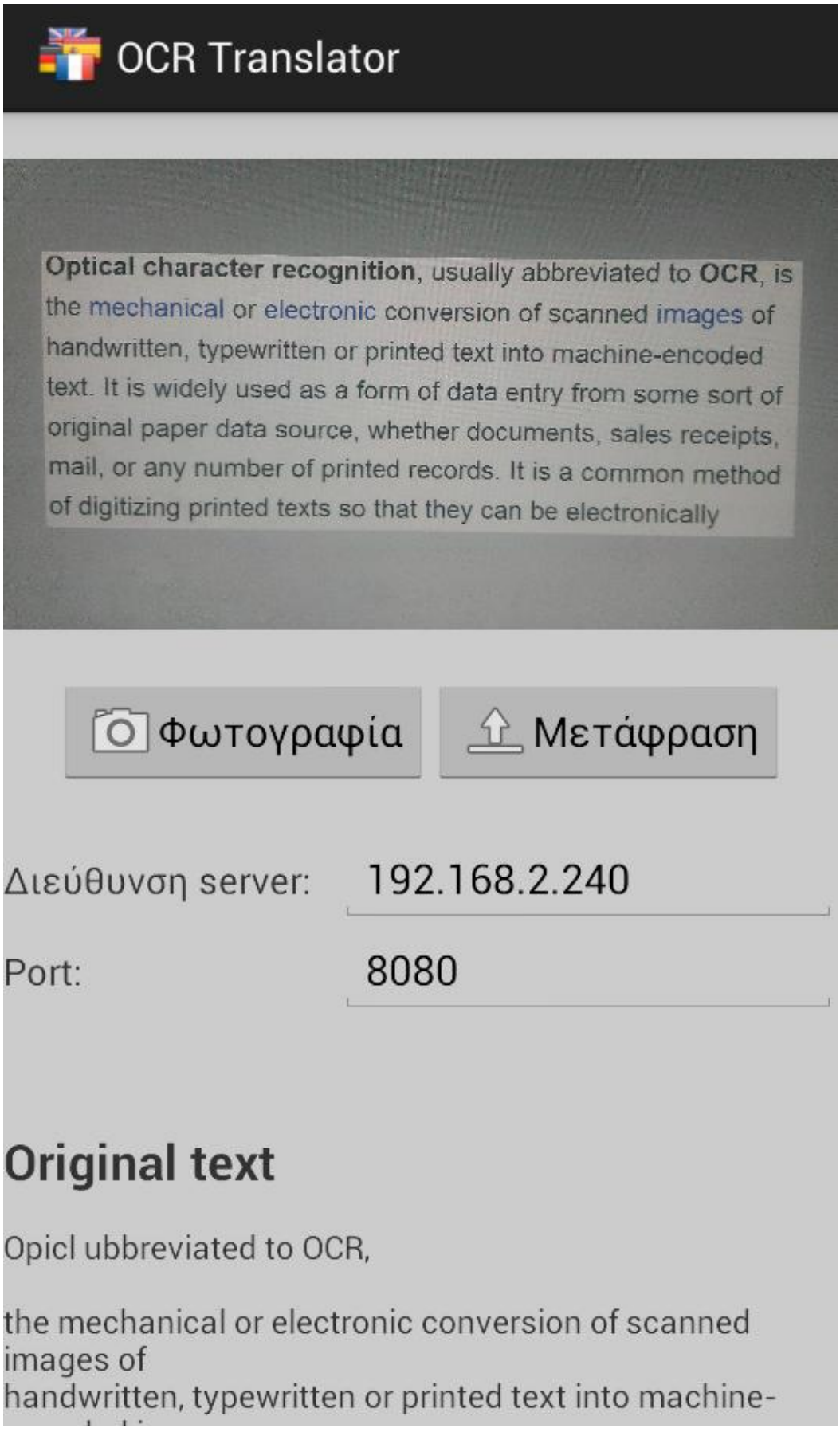


Η μετάφραση ξεκινάει πατώντας το κουμπί μετάφραση. Η εφαρμογή επικοινωνεί με τον server και εμφανίζει το αποτέλεσμα.



Για να εμφανιστεί ολόκληρο το αποτέλεσμα πρέπει να κάνει scroll προς τα κάτω.

Αν επιθυμεί να πάρει και άλλες φωτογραφίες τότε απλά χρειάζεται να πάει στο πάνω μέρος της οθόνης και να πατήσει το κουμπί φωτογραφία.


OCR Translator
Optical character recognition, usually abbreviated to OCR, is the mechanical or electronic conversion of scanned images of handwritten, typewritten or printed text into machine-encoded text. It is widely used as a form of data entry from some sort of original paper data source, whether documents, sales receipts, mail, or any number of printed records. It is a common method of digitizing printed texts so that they can be electronically
Φωτογραφία
Μετάφραση
Διεύθυνση server:
192.168.2.240
Port:
8080
Original text
Opicl ubbreviated to OCR,
the mechanical or electronic conversion of scanned images of
handwritten, typewritten or printed text into machine-

6 Συμπεράσματα

6.1 Γενικά

Οι εμπειρίες και οι παρατηρήσεις που είχαμε κατά τη συγγραφή της εργασίας διέφεραν πάρα πολύ από τις προσδοκίες μας. Επιλέξαμε να χρησιμοποιήσουμε τεχνολογίες από δύο τομείς, την οπτική αναγνώριση χαρακτήρων και την αυτόματη μετάφραση κειμένου οι οποίοι πριν από δύο δεκαετίες ήταν σε εμβρυακό στάδιο. Αν και παλιότερα η χρήση τέτοιων δυνατοτήτων σε έναν υπολογιστή για καθημερινές ανάγκες άγγιζε τα όρια της επιστημονικής φαντασίας, πλέον είναι αρκετά τετριμμένες και διαδεδομένες.

Για την ενσωμάτωση των παραπάνω δυνατοτήτων σε ένα πρόγραμμα χρειάστηκε να καταβάλλουμε ελάχιστη προσπάθεια, αφού είτε υπάρχουν έτοιμες σαν βιβλιοθήκες (στην περίπτωση του Tesseract) είτε προσφέρονται σαν δωρεάν υπηρεσίες (στην περίπτωση του Microsoft translate).

Ακόμη η χρήση μίας κινητής υπολογιστικής συσκευής, δηλαδή ενός κινητού τηλεφώνου, με το οποίο μπορούμε να παίρνουμε τις φωτογραφίες του κειμένου, τα τις στέλνουμε στον server και να παίρνουμε το αποτέλεσμα είναι αρκετά εύκολη υπόθεση πλέον, με τους κατασκευαστές να ενθαρρύνουν τους ανεξάρτητους προγραμματιστές να γράψουν τα δικά τους προγράμματα, προσφέροντάς τους βοήθεια και εργαλεία.

Το αποτέλεσμα ήταν πως, αντί να χρειαστεί να ασχοληθούμε με το κάθε ένα τμήμα της εφαρμογής ξεχωριστά, περάσαμε το μεγαλύτερο μέρος του χρόνου μας προσπαθώντας να συνδυάσουμε τα έτοιμα τμήματα ώστε να συνεργαστούν μεταξύ τους.

Παρακάτω αναφέρουμε συστηματικά τις παρατηρήσεις μας.

6.2 Παρατηρήσεις

Ξεκινώντας την εργασία μας έκανε εντύπωση το γεγονός ότι υπάρχουν αρκετές έτοιμες βιβλιοθήκες που επιτελούν τις εργασίες που ζητήσαμε. Σχετικά με την οπτική αναγνώριση χαρακτήρων υπάρχουν αρκετές επιλογές, αν και το Tesseract είναι πιο διαδεδομένο κυρίως λόγω της δυνατότητας που έχει να συνεργάζεται με προγράμματα γραμμένα σε πολλές γλώσσες. Μάλιστα υπήρχε η δυνατότητα να το συμπεριλάβουμε στην ίδια την εφαρμογή Android που δημιουργήσαμε, κάτι που απορρίψαμε λόγω των μεγάλων απαιτήσεων που έχει σε πόρους.

Επιπλέον το Tesseract, αν και θεωρείται πως δίνει αρκετά καλά αποτελέσματα σε σχέση με τα υπόλοιπα προγράμματα OCR, απέχει πολύ από το να είναι αποτελεσματικό στην αναγνώριση φωτογραφιών. Όταν η είσοδος είναι μία καθαρή εικόνα γενικά καταφέρνει να έχει συνεπή αποτελέσματα, όταν όμως πρόκειται για μία φωτογραφία οι αποκλείσεις είναι πάρα πολύ μεγάλες. Αυτό είναι ένα θέμα που έχουν όλες οι βιβλιοθήκες OCR και με το πέρασμα του χρόνου όσο πάει και βελτιώνεται.

Παρόλο που η χρήση του Tesseract είναι αρκετά εύκολη σε προγράμματα Java, όταν χρειάστηκε να το χρησιμοποιήσουμε σε συνδυασμό με τον Tomcat ανακαλύψαμε με έκπληξη ότι δεν υπάρχουν σχετικές οδηγίες. Η σωστή ρύθμιση του server ήταν ένα από τα πιο επίπονα κομμάτια της εφαρμογής.

Σχετικά με τη μετάφραση η υπηρεσία Microsoft Translate παρέχει εξίσου ανάμεικτα αποτελέσματα. Πρόκειται για έναν τομέα στον οποίο η έρευνα έχει επιταχυνθεί τα τελευταία χρόνια και όλες οι αντίστοιχες υπηρεσίες συνεχώς βελτιώνονται. Παρόλα αυτά,

αν και το αποτέλεσμα μίας μετάφρασης γενικά αρκεί για την εξαγωγή των γενικών νοημάτων του κειμένου, δεν μπορεί σε καμία περίπτωση να χρησιμοποιηθεί για περαιτέρω λειτουργίες.

Οι μεγαλύτερες εκπλήξεις προήρθαν από την έλλειψη δυνατοτήτων που θεωρούσαμε δεδομένες. Για παράδειγμα αν και το upload αρχείων από τον χρήστη είναι μία πολύ συνηθισμένη ενέργεια, δεν υποστηρίζεται άμεσα από τα servlets και χρειάστηκε η χρήση επιπλέον βιβλιοθηκών. Παρόλο που η διαδικασία για να γίνει αυτό επεξηγείται αρκετά καλά, κάποιος θα περίμενε πως η εγγενής υποστήριξη τέτοιων λειτουργιών θα ήταν αυτονόητη.

Στο ίδιο μήκος κύματος συναντήσαμε δυσκολίες με την αποστολή αρχείων με τις βιβλιοθήκες του Android μέσω του πρωτοκόλλου HTTP. Εξαιτίας της τεράστιας διάδοσης αυτού του standard περιμέναμε να υποστηρίζεται και αυτό εγγενώς, κάτι που δεν ισχύει. Εν τέλει χρησιμοποιήσαμε τη βιβλιοθήκη Retrofit, η οποία αναλαμβάνει να συγχρονίσει αυτόματα και τα διαφορετικά threads που πρέπει να υπάρχουν για την επικοινωνία, ένας τομέας που και πάλι είναι αρκετά πολύπλοκος με τις βασικές βιβλιοθήκες.

Ειδικά σχετικά με το Android η κατάσταση του documentation είναι σχετικά άσχημη. Σε αυτό συμβάλλουν οι πολλές διαφορετικές εκδόσεις, όμως ακόμη και στο επίσημο site είδαμε παραδείγματα κώδικα που είτε δεν είχαν ανανεωθεί για να αντικατοπτρίζουν τις αλλαγές στην τελευταία έκδοση του API, είτε ήταν ελλιπή.

6.3 Μελλοντική επέκταση

Αν και η εργασία επιτελεί το στόχο που είχαμε βάλει υπάρχουν κάποιοι τομείς στους οποίους θα μπορούσε μελλοντικά να δοθεί βάρος.

Ο πρώτος έχει να κάνει με την ποιότητα των αποτελεσμάτων τόσο της αναγνώρισης χαρακτήρων όσο και της μετάφρασης. Αν και επιλέξαμε τις λύσεις που θεωρήσαμε ότι ήταν οι περισσότερο ενδεδειγμένες θα είχε ενδιαφέρον η χρήση διαφορετικών εργαλείων ώστε να μπορεί να γίνει σύγκριση της ακρίβειας των νέων αποτελεσμάτων με αυτά που βρήκαμε.

Μία εναλλακτική υλοποίηση θα μπορούσε να ενσωματώνει τις βιβλιοθήκες OCR και μετάφρασης στην ίδια την εφαρμογή του κινητού, καθιστώντας περιττή την ύπαρξη του server. Αν και αποφύγαμε αυτή τη λύση λόγω των περιορισμένων πόρων κάποιων συσκευών, η εμφάνιση συνεχώς και δυνατότερων τηλεφώνων κάνει μια τέτοια λύση να φαίνεται όλο και περισσότερο εφικτή.

7 Βιβλιογραφία

About Optical Character Recognition in Google Drive. [Ηλεκτρονικό] https://support.google.com/drive/answer/176692?hl=en.

Android Architecture – The Key Concepts of Android OS. *www.android-app-market.com.* [Ηλεκτρονικό] http://www.android-app-market.com/android-architecture.html.

Android Camera API - Tutorial. [Ηλεκτρονικό] http://www.vogella.com/tutorials/AndroidCamera/article.html.

Android Developers. *android.com.* [Ηλεκτρονικό] http://developer.android.com/index.html.

Android NDK. *developer.android.com.* [Ηλεκτρονικό] http://developer.android.com/tools/sdk/ndk/index.html.

Android working with Camera. [Ηλεκτρονικό] http://www.androidhive.info/2013/09/android-working-with-camera-api/.

Building the Tesseract NDK library for Android. *Gabriel Novak – Florida Atlantic University – Computer Science.* [Ηλεκτρονικό] http://droidcomp.wordpress.com/2012/08/04/building-the-tesseract-ndk-library-for-android/.

Camera. [Ηλεκτρονικό] http://developer.android.com/guide/topics/media/camera.html.

Definitions of Smartphone. *Pc Magazine Encyclopedia.* [Ηλεκτρονικό] http://www.pcmag.com/encyclopedia_term/0,2542,t=Smartphone&i=51537,00.asp#fbid=PsmZ.

Developer Tools. *developer.android.com.* [Ηλεκτρονικό] http://developer.android.com/tools/index.html.

Get the Android SDK. *developer.android.com.* [Ηλεκτρονικό] http://developer.android.com/sdk/index.html.

iPhone. *Βικιπαίδεια.* [Ηλεκτρονικό] http://el.wikipedia.org/wiki/IPhone.

Leptonica . *leptonica.com.* [Ηλεκτρονικό] http://leptonica.com/.

Machine translation. [Ηλεκτρονικό] http://en.wikipedia.org/wiki/Machine_translation.

Setting up Automatic NDK Builds in Eclipse. *mobilepearls.com.* [Ηλεκτρονικό] http://mobilepearls.com/labs/ndk-builder-in-eclipse/.

Taking Photos Simply. [Ηλεκτρονικό] http://developer.android.com/training/camera/photobasics.html.

Tesseract (software). *wikipedia.org.* [Ηλεκτρονικό] http://en.wikipedia.org/wiki/Tesseract_%28software%29.

Tesseract full integration . [Ηλεκτρονικό] http://forums.alfresco.com/forum/installation-upgrades-configuration-integration/integration-other-systems/tesseract-full.

Tesseract OCR. [Ηλεκτρονικό] http://sourceforge.net/projects/tesseract-ocr/.

tesseract-ocr. *code.google.com.* [Ηλεκτρονικό] http://code.google.com/p/tesseract-ocr/.

tess-two. *github.com.* [Ηλεκτρονικό] https://github.com/rmtheis/tess-two.

Using Tess4J on Tomcat. [Ηλεκτρονικό] http://stackoverflow.com/questions/11736486/using-tess4j-on-tomcat.

What is automated translation. [Ηλεκτρονικό] http://www.sdl.com/technology/language-technology/what-is-automated-translation.html.

Έξυπνο τηλέφωνο. *Βικιπαίδεια.* [Ηλεκτρονικό] http://el.wikipedia.org/wiki/%CE%88%CE%BE%CF%85%CF%80%CE%BD%CE%BF_%CF%84%CE%B7%CE%BB%CE%AD%CF%86%CF%89%CE%BD%CE%BF.

Μάθε τα πάντα για Windows Phone 8. *techblog.gr.* [Ηλεκτρονικό] http://techblog.gr/tag/windows-phone-8/.

Μηλιαράς, Κωνσταντίνος. 2011. Ηλεκτρονικός τουριστικός Οδηγός σε έκδοση για έξυπνα. 2011.

Ορολογία του Android. *myphone.gr.* [Ηλεκτρονικό] http://www.myphone.gr/forum/showthread.php?t=306146.

Σουίτα ασφαλείας και anti-virus Sophos Mobile Security 3.0 για Android συσκευές . *nss.gr.* [Ηλεκτρονικό] https://www.nss.gr/news/445-latest-free-version-of-sophos-mobile-security-delivers-application-protection-faster-scanning-and-web-protection.html?format=html&lang=el.

Τα «έξυπνα» κινητά τηλέφωνα πωλούν περισσότερο από τα συμβατικά. *kathimerini.gr.* [Ηλεκτρονικό] http://www.kathimerini.gr/4dcgi/_w_articles_kathremote_1_16/08/2013_514236.

8 Παράρτημα

Παρακάτω παραθέτουμε τον κώδικα της εφαρμογής.

8.1 Web application

8.1.1 Κλάση ProcessImage

```
package gr.teiher.epp.ocrtranslator;

import java.awt.image.BufferedImage;
import java.io.IOException;
import java.io.InputStream;
import java.io.PrintWriter;
import java.util.Iterator;
import java.util.List;

import javax.imageio.ImageIO;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import net.sourceforge.tess4j.Tesseract;
import net.sourceforge.tess4j.TesseractException;

import org.apache.commons.fileupload.FileItem;
import org.apache.commons.fileupload.FileItemFactory;
import org.apache.commons.fileupload.FileUploadException;
import org.apache.commons.fileupload.disk.DiskFileItemFactory;
import org.apache.commons.fileupload.servlet.ServletFileUpload;
import org.json.simple.JSONObject;

import com.memetix.mst.detect.Detect;
import com.memetix.mst.language.Language;
import com.memetix.mst.translate.Translate;

/**
 * Servlet implementation class ProcessImage
 */
@WebServlet("/ProcessImage")
public class ProcessImage extends HttpServlet {
	private static final long serialVersionUID = 1L;

	public ProcessImage() {
		super();

	}

	protected void doGet(HttpServletRequest request,
			HttpServletResponse response) throws
ServletException, IOException {

	}
```

```
      protected void doPost(HttpServletRequest request,
                  HttpServletResponse response) throws
ServletException, IOException {
            request.setCharacterEncoding("UTF-8");
            response.setCharacterEncoding("UTF-8");
            response.setContentType("application/json");

            ImageIO.scanForPlugins();
            PrintWriter out = response.getWriter();

            if (!ServletFileUpload.isMultipartContent(request)) {
                  return;
            }

            FileItemFactory factory = new DiskFileItemFactory();
            ServletFileUpload upload = new ServletFileUpload(factory);

            InputStream inputStream = null;

            String originalText = "OCR FAILED";
            String translatedText = "TRANSLATION FAILED";

            JSONObject obj = new JSONObject();

            try {
                  List items = upload.parseRequest(request);

                  Iterator iterator = items.iterator();
                  while (iterator.hasNext()) {
                        FileItem item = (FileItem) iterator.next();

                        if (!item.isFormField()) {
                              inputStream = item.getInputStream();
                              break;
                        }
                  }

                  // Tesseract1 instance = new Tesseract1();
                  Tesseract instance = Tesseract.getInstance();
                  String tessdataPath =
getServletContext().getRealPath(
                              "/WEB-INF/lib/tessdata/");
                  System.out.println("==> " + tessdataPath);
                  instance.setDatapath(tessdataPath);

                  BufferedImage image = ImageIO.read(inputStream);

                  originalText = instance.doOCR(image);

                  Translate.setClientId("OCR_SERVER");
                  Translate

      .setClientSecret("oCb8El1QYc3ta6QKTawyOidKs1CGE/vVm7Y9oMwQPh8=");

                  Language originalLanguage = null;
                  Language transationLanguage = Language.GREEK;

                  originalLanguage = Detect.execute(originalText);
```

```
                    System.out.println("==> " + originalLanguage);
                    translatedText = Translate.execute(originalText,
originalLanguage,
                                  Language.GREEK);
                    System.out.println("===> " + translatedText);

                    obj.put("originalLanguage",
originalLanguage.toString());

             } catch (FileUploadException e) {
                    System.err.println("Error during upload");
                    e.printStackTrace();
             } catch (TesseractException e) {
                    System.err.println("OCR error");
                    e.printStackTrace();
             } catch (Exception e) {
                    e.printStackTrace();
             }

             obj.put("originalText", originalText);
             obj.put("translatedText", translatedText);
             out.print(obj);

      }

}
```

8.1.2 Αρχείο uploadForm.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
      <head>
            <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
            <title>Επεξεργασία εικόνας</title>
      </head>
      <body>
            <form method="post" action="ProcessImage"
enctype="multipart/form-data">
                    <input type="file" name="imageFile" />

                    <br />

                    <input type="submit" value="Επεξεργασία">
             </form>
      </body>
</html>
```

8.2 Android application

8.2.1 Αρχείο strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">OCR Translator</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
    <string name="take_photo">Φωτογραφία</string>
    <string name="translate">Μετάφραση</string>
    <string name="server_address">Διεύθυνση server:</string>
    <string name="server_port">Port:</string>

</resources>
```

8.2.2 Αρχείο AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="gr.teiher.epp.ocrtranslator.app"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="10"
        android:targetSdkVersion="19" />

      <uses-feature android:name="android.hardware.camera"
                  android:required="true" />

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"
/>
      <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="gr.teiher.epp.ocrtranslator.app.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
```

```
/>
            </intent-filter>
        </activity>
    </application>

</manifest>
```

8.2.3 Αρχείο activity_main.xml

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/scrollView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#cccccc" >

    <LinearLayout
        android:id="@+id/container"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:descendantFocusability="beforeDescendants"
        android:focusableInTouchMode="true"
        android:orientation="vertical" >

        <ImageView
            android:id="@+id/photoView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="20dp"
            android:adjustViewBounds="true"
            android:src="@android:drawable/ic_menu_report_image"
            tools:ignore="ContentDescription" />

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="20dp"
            android:layout_marginTop="20dp"
            android:gravity="center_horizontal" >

            <Button
                android:id="@+id/photoButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:drawableLeft="@android:drawable/ic_menu_camera"
                android:text="@string/take_photo" />

            <Button
                android:id="@+id/translateButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:drawableLeft="@android:drawable/ic_menu_upload"
                android:text="@string/translate" />
```

```
        </LinearLayout>


        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content" >

            <TextView
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_gravity="center_vertical"
                android:layout_weight="0.4"
                android:text="@string/server_address"
                android:textSize="16sp" />

            <EditText
                android:id="@+id/serverAddressText"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_gravity="center_vertical"
                android:layout_weight="0.6" />
        </LinearLayout>


        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content" >

            <TextView
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_gravity="center_vertical"
                android:layout_weight="0.4"
                android:text="@string/server_port"
                android:textSize="16sp" />

            <EditText
                android:id="@+id/serverPortText"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_gravity="center_vertical"
                android:layout_weight="0.6"
                android:inputType="number" />
        </LinearLayout>


        <ProgressBar
            android:id="@+id/loadingBar"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center_horizontal"
            android:visibility="invisible" />

        <TextView
            android:id="@+id/responseText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textSize="14sp" />
```

```
    </LinearLayout>

</ScrollView>
```

8.2.4 Κλάση Config

```
package gr.teiher.epp.ocrtranslator.app;

public class Config {
	public static final String TAG = "OCR";

	public static final String DEFAULT_SERVER_ADDRESS =
"192.168.2.240";
	public static final String DEFAULT_SERVER_PORT = "8080";

	public static final String IMAGE_FILE_NAME = "IMAGE";
	public static final String IMAGE_FILE_EXT = ".jpg";

}
```

8.2.5 Κλάση OcrData

```
package gr.teiher.epp.ocrtranslator.app;

import java.io.Serializable;

import com.google.gson.annotations.SerializedName;

public class OcrData implements Serializable{

	@SerializedName("originalText")
	private String originalText;

	@SerializedName("originalLanguage")
	private String originalLanguage;

	@SerializedName("translatedText")
	private String translatedText;

	public OcrData() {

	}

	public String getOriginalText() {
		return originalText;
	}

	public void setOriginalText(String originalText) {
```

```
		this.originalText = originalText;
	}

	public String getOriginalLanguage() {
		return originalLanguage;
	}

	public void setOriginalLanguage(String originalLanguage) {
		this.originalLanguage = originalLanguage;
	}

	public String getTranslatedText() {
		return translatedText;
	}

	public void setTranslatedText(String translatedText) {
		this.translatedText = translatedText;
	}


}
```

8.2.6 Interface OcrService

```
package gr.teiher.epp.ocrtranslator.app;

import retrofit.Callback;
import retrofit.http.Multipart;
import retrofit.http.POST;
import retrofit.http.Part;
import retrofit.mime.TypedFile;

public interface OcrService {
	@Multipart
	@POST("/OCR_Translator/ProcessImage")
	void sendImage(@Part("imageFile") TypedFile photo,
Callback<OcrData> callback);
}
```

8.2.7 Κλάση MainActivity

```
package gr.teiher.epp.ocrtranslator.app;


import java.io.File;
import java.io.IOException;

import retrofit.Callback;
```

```
import retrofit.RestAdapter;
import retrofit.RetrofitError;
import retrofit.client.Response;
import retrofit.mime.TypedFile;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.support.v7.app.ActionBarActivity;
import android.text.Html;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;
import static gr.teiher.epp.ocrtranslator.app.Config.TAG;

public class MainActivity extends ActionBarActivity {
	public static final int REQUEST_IMAGE_CAPTURE = 0x1;

	private RestAdapter restAdapter;

	private boolean pictureTaken = false;
	private File photoFile = null;
	private String photoFilePath = null;

	private ImageView photoView;
	private EditText serverAddressText, serverPortText;
	private TextView responseText;
	private ProgressBar loadingBar;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);

		photoView = (ImageView) findViewById(R.id.photoView);
		loadingBar = (ProgressBar) findViewById(R.id.loadingBar);
		responseText = (TextView) findViewById(R.id.responseText);

		serverAddressText = (EditText)
findViewById(R.id.serverAddressText);
		serverAddressText.setText(Config.DEFAULT_SERVER_ADDRESS);

		serverPortText = (EditText)
findViewById(R.id.serverPortText);
		serverPortText.setText(Config.DEFAULT_SERVER_PORT);

		Button photoButton = (Button)
findViewById(R.id.photoButton);
		photoButton.setOnClickListener(new View.OnClickListener() {

			@Override
```

```
                    public void onClick(View v) {

                        try {
                           photoFile =
File.createTempFile(Config.IMAGE_FILE_NAME, Config.IMAGE_FILE_EXT,
getExternalCacheDir());
                           photoFilePath = "file:" +
photoFile.getAbsolutePath();
                           responseText.setText("");

                           Intent takePictureIntent = new
Intent(MediaStore.ACTION_IMAGE_CAPTURE);

      takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT,
Uri.fromFile(photoFile));

      startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);

                           } catch (IOException e) {
                                 Toast.makeText(MainActivity.this, "Error
creating image file.", Toast.LENGTH_SHORT).show();;
                                 e.printStackTrace();
                           }


                    }
             });

             Button translateButton = (Button)
findViewById(R.id.translateButton);
             translateButton.setOnClickListener(new
View.OnClickListener() {

                    @Override
                    public void onClick(View v) {
                           if(!pictureTaken) {
                                 Toast.makeText(MainActivity.this,
"Please take a picture.", Toast.LENGTH_SHORT).show();

                           } else {
                           responseText.setText("");
                           loadingBar.setVisibility(View.VISIBLE);
                                 sendImage();

                           }

                    }
             });

      }

      @Override
      protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
          if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode ==
RESULT_OK) {
            Log.i(TAG, "photoFilePath:" + photoFilePath);
            Log.i(TAG, "photoFile:" + photoFile + ", " +
```

```
photoFile.length());

            Bitmap imageBitmap = resizeImage();

            Log.i(TAG, "imageBitmap:" + imageBitmap);

                    photoView.setImageBitmap(imageBitmap);
                    pictureTaken = true;

           }

      }

      private Bitmap resizeImage() {
             BitmapFactory.Options bmOptions = new
BitmapFactory.Options();
             bmOptions.inJustDecodeBounds = true;
             BitmapFactory.decodeFile(photoFile.getAbsolutePath(),
bmOptions);
             int photoW = bmOptions.outWidth;
             int photoH = bmOptions.outHeight;

             float ratio = (float)photoW / photoH;

             int targetW = photoView.getWidth();
             int targetH = (int)(targetW * ratio);

             int scaleFactor = Math.min(photoW/targetW, photoH/targetH);


             bmOptions.inJustDecodeBounds = false;
             bmOptions.inSampleSize = scaleFactor;
             bmOptions.inPurgeable = true;

             Bitmap bitmap =
BitmapFactory.decodeFile(photoFile.getAbsolutePath(), bmOptions);

             return bitmap;

      }

      public void sendImage() {
             String serverAddress =
                          "http://"
                          + serverAddressText.getText()
                          + ":"
                          + serverPortText.getText();

             restAdapter = new RestAdapter.Builder()
                          .setEndpoint(serverAddress)
                          .build();

             OcrService ocrService =
restAdapter.create(OcrService.class);
             TypedFile f = new TypedFile("image/jpeg", photoFile);
             ocrService.sendImage(f, new Callback<OcrData>() {

                    @Override
```

```
                    public void success(OcrData data, Response resp) {
                          loadingBar.setVisibility(View.INVISIBLE);


      data.setOriginalText(data.getOriginalText().replaceAll("\n",
"<br>"));

      data.setTranslatedText(data.getTranslatedText().replaceAll("\n",
"<br>"));

                          String res = "<h1>Original text</h1><p>" +
data.getOriginalText() + "</p>";
                          res += "<h1>Language: " +
data.getOriginalLanguage() +"</h1>";
                          res += "<h1>Translation</h1><p>" +
data.getTranslatedText() + "</p>";

                          responseText.setText(Html.fromHtml(res));
                    }

                    @Override
                    public void failure(RetrofitError err) {
                          loadingBar.setVisibility(View.INVISIBLE);
                          Toast.makeText(MainActivity.this, "Error
connecting to servere.", Toast.LENGTH_SHORT).show();
                          err.printStackTrace();
                    }
              });
      }

}
```