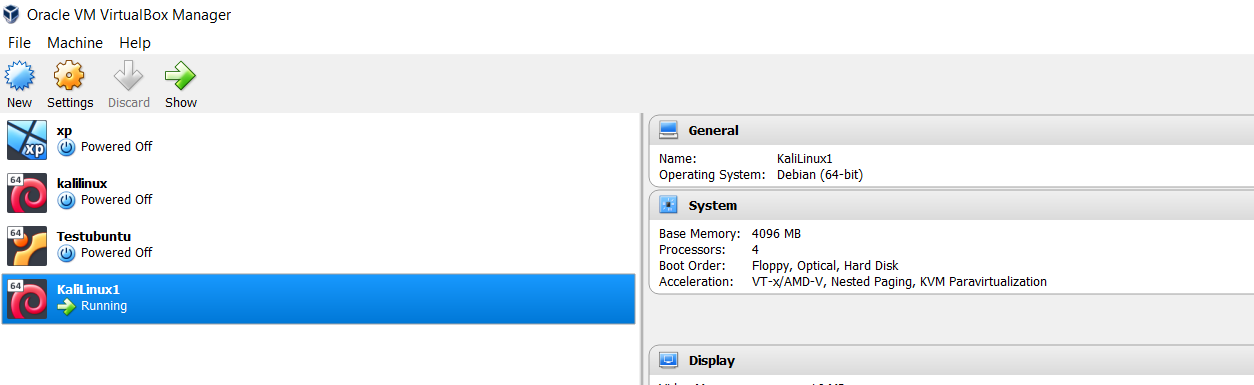


|  |  |
| --- | --- |
| **Στοιχεία Φοιτητή** | |
| Επώνυμο Φοιτητή | Κουσουννής |
| Όνομα Φοιτητή | Κωνσταντίνος |
| Αριθμός Μητρώου | Π14086 |
| Email | Kwstas654321@gmail.com |

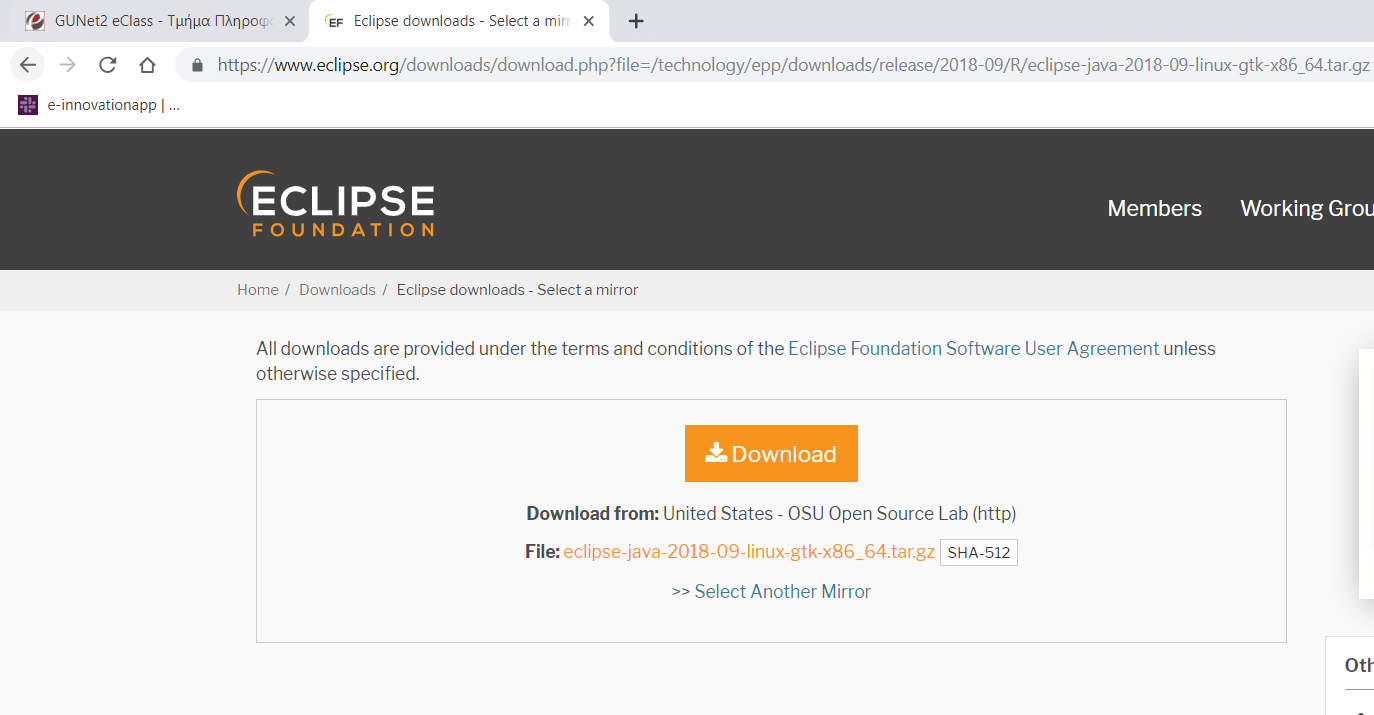
**Ανάλυση Κώδικα**

Η μηχανή αναζήτησης έχει δημιουργηθεί σε λειτουργικό **Kali Linux** το οποίο δημιουργήσαμε μέσα από το **Virtual box** .



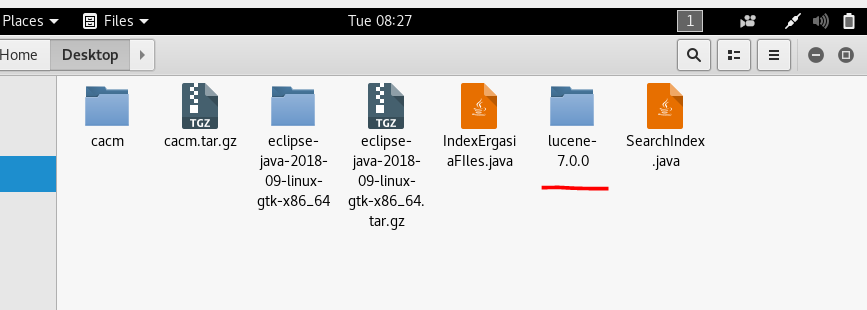
Στο λειτουργικό μας κατεβάσαμε το Eclipse IDE από την ιστοσελίδα

https://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/2018-09/R/eclipse-java-2018-09-linux-gtk-x86\_64.tar.gz



Στην συνέχεια κατεβάζουμε τις βιβλιοθήκες της Lucene την αντίστοιχη έκδοση από την επίσημη ιστοσελίδα της lucene.

<https://lucene.apache.org/core/downloads.html>

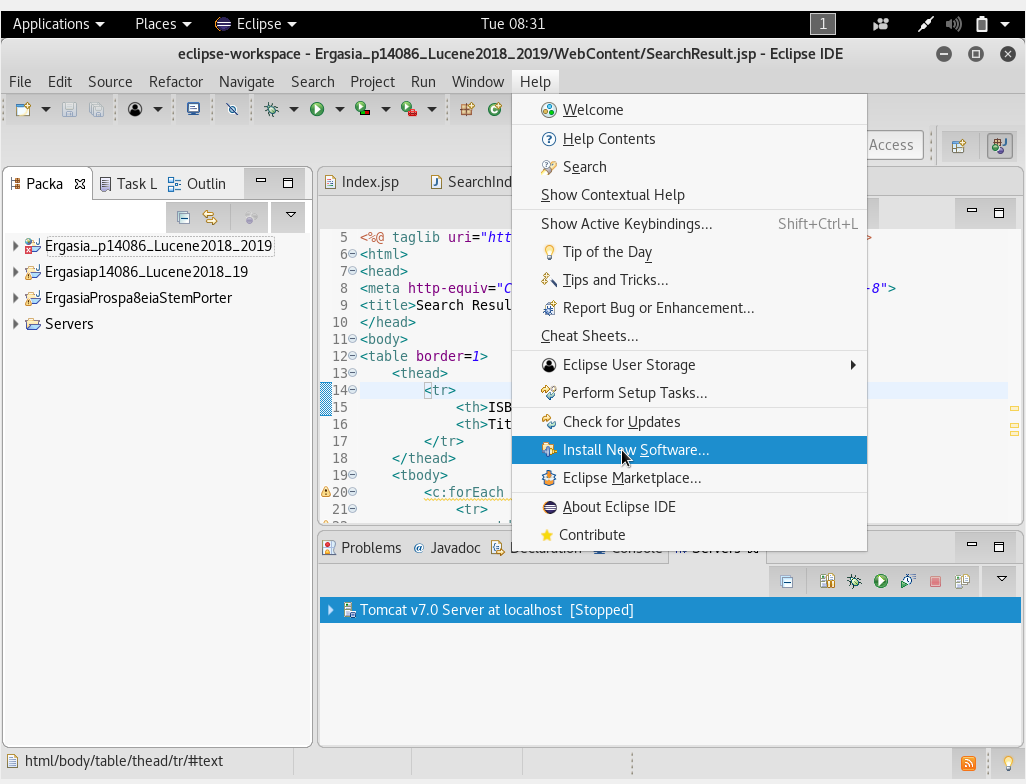


Η δικιά μου έκδοση είναι στην Lucene 7.0.0

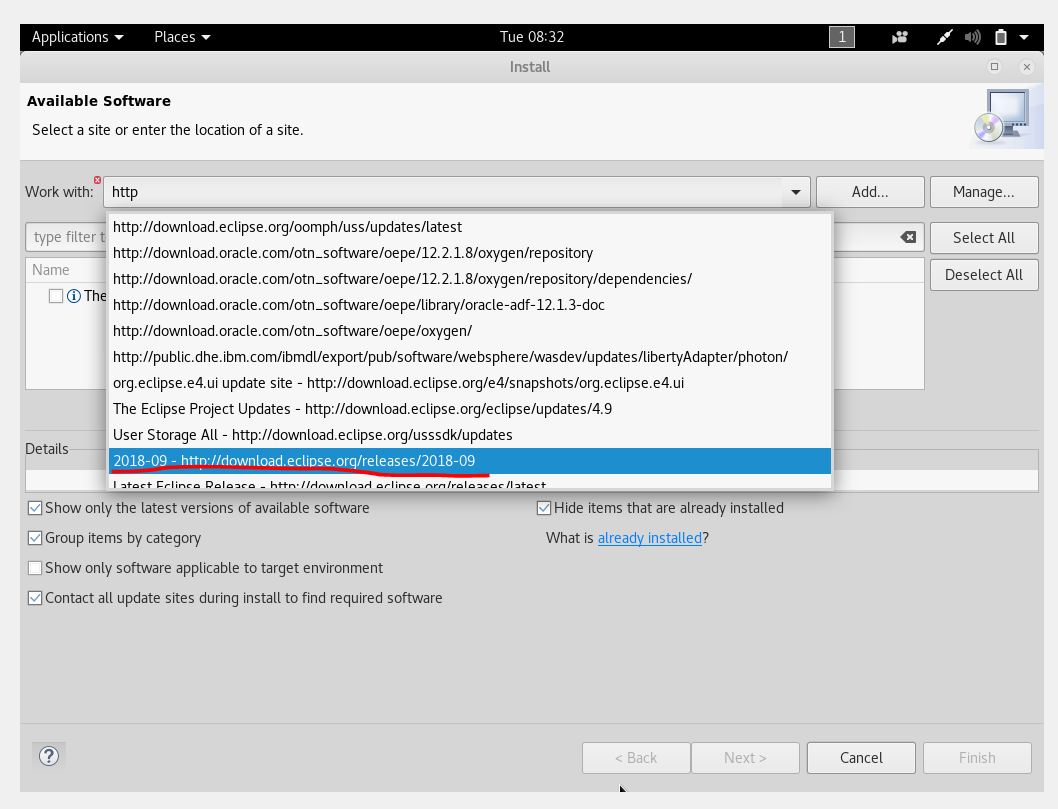
**Δημιουργία Dynamic Web Project**

Στο περιβάλλον eclipse IDE επιλέγουμε να εγκαταστήσουμε το java EE

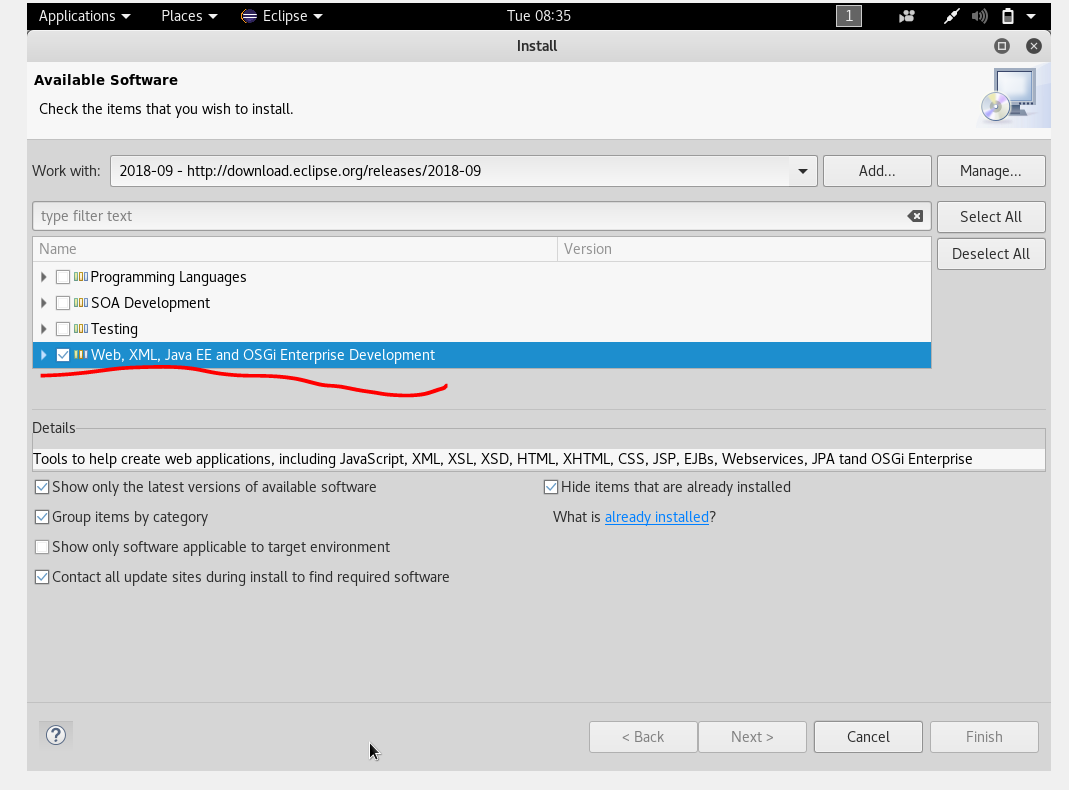
Επιλέγουμε HELP->Install New Software



Στην συνέχεια πληκτρολογούμε **http://** και επιλέγουμε το 2018-2019

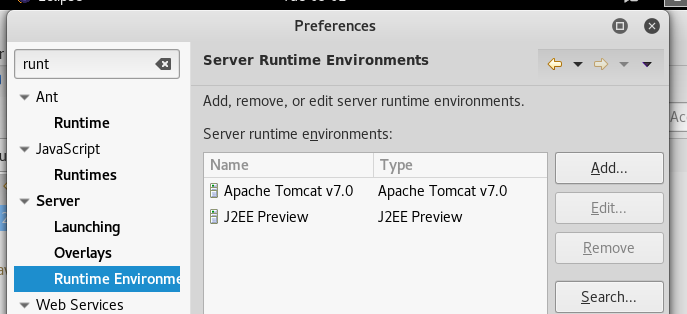


Στην συνέχεια κάνουμε εγκατάσταση τα πακέτα Web,XML,Java EE

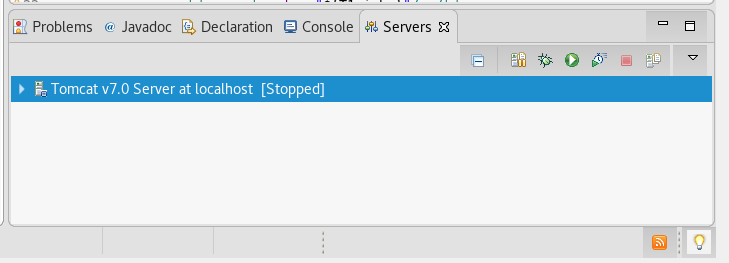


Στην συνέχεια δημιουργούμε έναν Server για να τρέξουμε το πρόγραμμα μας .

Επιλέγουμε Window->Preferences ->Runtime Environment και επιλέγουμε την αντίστοιχη έκδοση του apache που έχουμε.

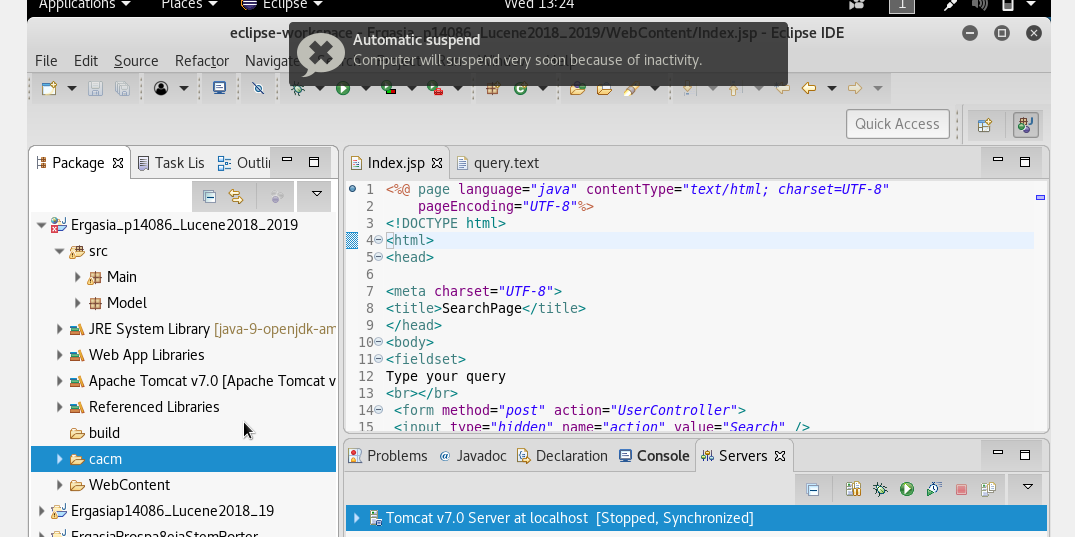


Εμείς εδώ βάλλαμε Tomcat 7.0.0



Οπότε επιλέγουμε να κάνουμε import το Project που έχουμε επιλέγουμε

**File-> New -> Other->Dynamic Web Project ->Ergasia\_p14086\_Lucene2018\_2019**

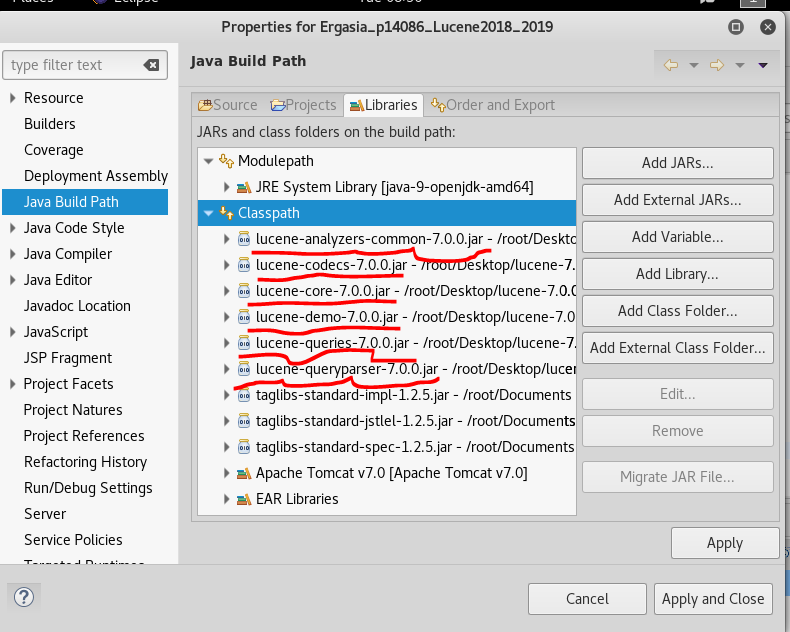


Αφού δημιουργήσουμε θέλουμε να κάνουμε import της βιβλιοθήκες της Lucene.

**Εισαγωγή Βιβλιοθήκης Lucene**

Αφού δημιουργήσουμε το project μας πατάμε δεξί κλικ **build path->configure build path**

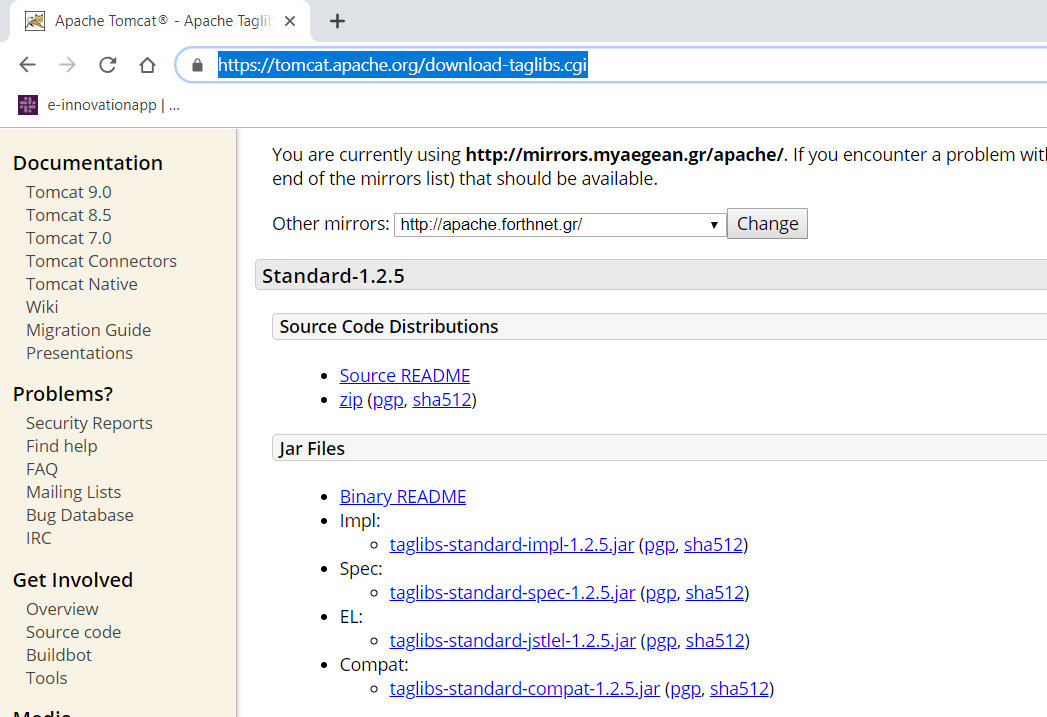
**Add external jars** και επιλέγουμε τα αντίστοιχα jars.



Επιλέγουμε να κάνουμε Add external jars . Από τα jars της βιβλιοθήκης Lucene που κατεβάσαμε στην αρχή .

Επιπλέον κατεβάζουμε από την ιστοσελίδα του apache tomcat jstl

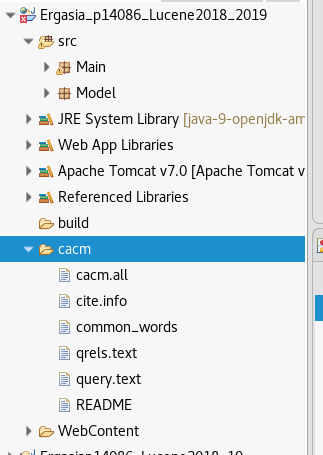
[**https://tomcat.apache.org/download-taglibs.cgi**](https://tomcat.apache.org/download-taglibs.cgi)



Θέλουμε αυτά τα πακέτα για να τρέχουμε jstl κώδικα για να εμφανίζουμε στην ιστοσελίδα μας δυναμικούς πίνακες για να βλέπουμε τα κείμενα μας το καθένα ξεχωριστά .

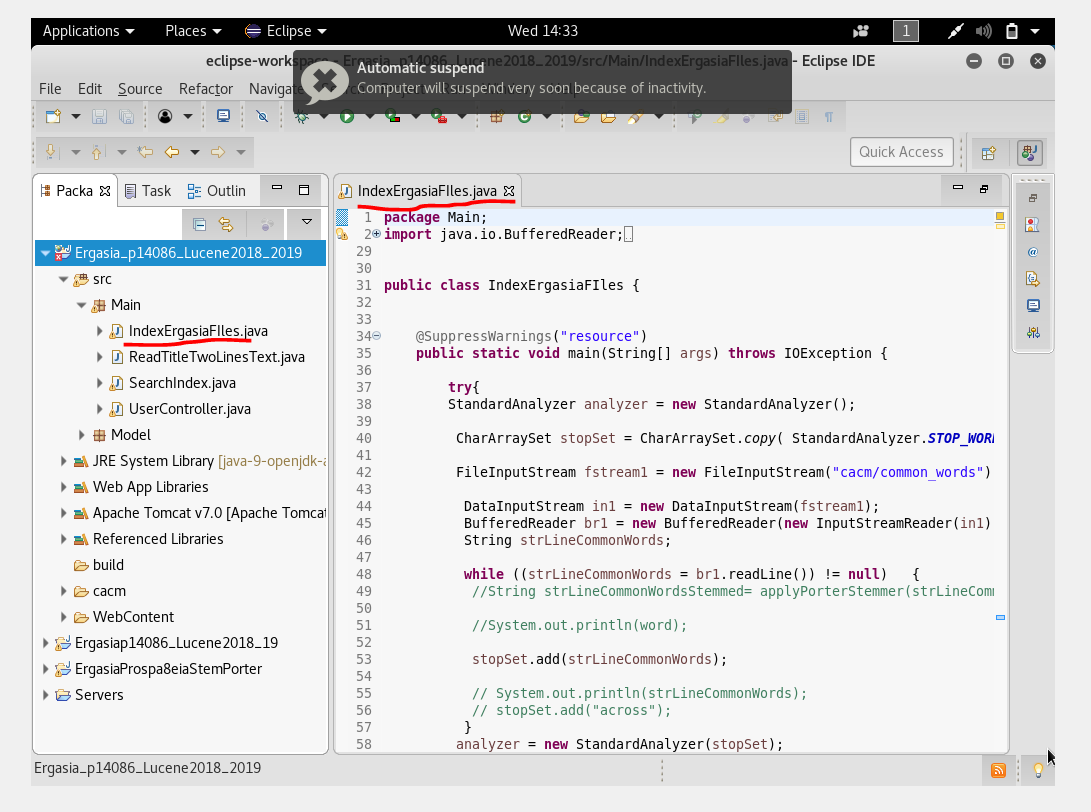
**Εισαγωγή Φακέλου Cacm από εκφώνηση άσκησης**

Κατεβάζουμε από την ιστοσελίδα του μαθήματος το αντίστοιχο διαθέσιμο αρχείο και το κάνουμε αντιγραφή μέσα στο project μας . Με αυτό τον τρόπο θα διαβάσουμε το αρχείο και θα δημιουργήσουμε τα ευρετήρια μας με την βοήθεια της βιβλιοθήκης Lucene .Επιπλέον θα χρησιμοποιήσουμε το αρχείο common\_words το οποίο και θα διαβάσουμε για να αγνοήσουμε της ποιο συνηθισμένες λέξεις στο αρχείο μας . Επιπλέον θα χρησιμοποιήσουμε το query.txt για να πάρουμε της ερώτησης που θα κάνουμε στο σύστημα μας .Τέλος θα χρησιμοποιήσουμε qrels.txt για να δούμε πια κείμενα είναι σχετικά και πια όχι για να κάνουμε τα διαγράμματα precision-recall.



**Δημιουργία IndexErgasiaFiles.java**

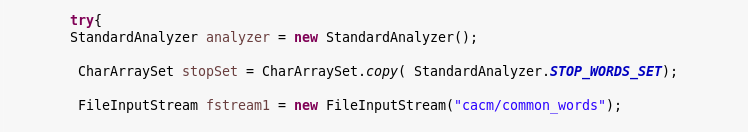
Επιλέγουμε πάνω στο Project μας δεξί κλικ ->new ->class και δημιουργούμε το αρχείο μας με ονομασία **IndexErgasiaFiles.java**



Για αυτό το αρχείο java πληκτρολογούμε στον κώδικα

**Public static void main**

To αρχείο αυτό θα τρέχει ξεχωριστά από την ιστοσελίδα μας. Θα τρέχουμε το αρχείο μόνο μια φορά για να δημιουργήσουμε τα ευρετήρια μας αν τρέξουμε το αρχείο παραπάνω από μια φορά θα έχουμε ευρετήρια με διπλότυπα στοιχεία .



StandardAnalyzer analyzer =new StandardAnalyzer ();

Ορίζουμε το analyzer για να κάνουμε με την βοήθεια της Lucene ανάλυση των κειμένων σε λέξεις .Επιλογή των όρων (λέξεων κλειδιών) που θα αποθηκευτούν στο ευρετήριο μας

CharArraySet stopSet=CharArraySet.copy(StandardAnalyzer.**STOP\_WORDS\_SET**)

Πέρα από τον analyzer θέλουμε να αγνοούμε της λέξεις που έχουμε στο αρχείο common\_words μέσα στον φάκελο cacm . Χρησιμοποιούμε την κλάση CharArraySet η οποία θα πάρει στην συνέχεια της λέξεις του αρχείου common\_words. Αυτή η κλάση της Lucene παραβλέπει λέξεις οι οποίες χρησιμοποιούνται συχνά σε όλα τα κείμενα



**FileInputStream fstream1=new FileInputStream(“cacm/common\_words”)**

Δηλώνουμε το αντίστοιχο path που βρίσκεται το αρχείο που θέλουμε να διαβάσουμε .Χρειαζόμαστε το αρχείο **common\_words**.

**BufferedReader br1=new BufferedReader (new InputStreameReader(in1))**

Δηλώνουμε ότι θέλουμε μόνο να διαβάσουμε το αρχείο μας.

String strLineCommonWords;

Δήλωση αλφαριθμητικού για να παίρνουμε τα δεδομένα από το αρχείο γραμμή γραμμή.

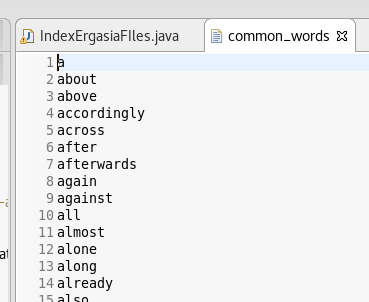
while ((strLineCommonWords = br1.readLine()) != null) {

stopSet.add(applyPorterStemmer(strLineCommonWords));

}

Δημιουργία επανάληψης για να διαβάζουμε το αρχείο μας γραμμή προς γραμμή.

Το αρχείο μας περιέχει σε κάθε γραμμή και από μια λέξη



Από εκφώνηση ξέρουμε ότι θέλουμε να χρησιμοποιούμε τον Porter αλγόριθμο για Stemming οπότε όλο το κείμενα μας θα δούμε στην συνέχεια ότι τα έχουμε περάσει στο σύστημα μας μέσα από αυτόν τον αλγόριθμο .Άρα για να αγνοήσουμε τις λέξεις μας θα πρέπει να περάσουμε της λέξης μέσα από τον Αλγόριθμο Porter για Stemming.

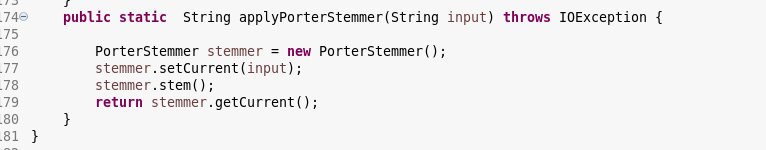
**stopSet.add (applyPorterStemmer (strLineCommonWords))**

οπότε καλώ την κλάση που έχω δημιουργήσει τον αλγόριθμο του Porter και στην συνέχεια περνάω την λέξη κάθε φορά μέσα στην Lucene για να αγνοήσει τις λέξεις.

**stopSet.add**

**Κλάση ApplyPorterStemmer**





Αρχικά δηλώνουμε την βιβλιοθήκη του Porter Stemmer

Η κλάση αυτή βρίσκεται στο κάτω μέρος του κώδικα μας ,παίρνει ως όρισμα ένα αλφαριθμητικό

**String input** ο αλγόριθμος εφαρμόζεται σε μία λέξη όποτε δεν γίνεται να του δώσουμε να μας μετατρέψει προτάσεις αλλά πραγματοποιείτε ανά λέξη.

Porter Stemmer stemmer = new Porter Stemmer ();

Καλούμε την κλάση Porter Stemmer από τις βιβλιοθήκες που κάναμε add.

Stemmer.setCurrent(input);

Παίρνουμε την λέξη

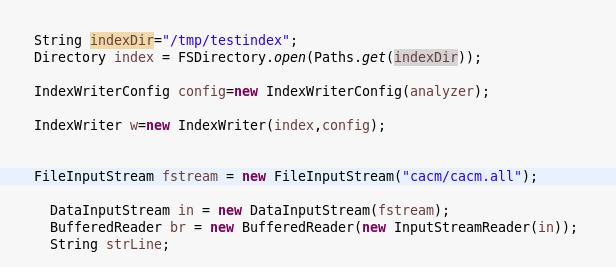
Stemmer.stem ();

Εδώ βρίσκουμε το ριζικό

return stemmer.getCurrent();

Επιστρέφουμε το ριζικό μας δηλαδή έχουμε εφαρμόσει τον αλγόριθμο του Porter για Stemming.

**Δημιουργία Ευρετηρίων**



Θέτουμε το path που θέλουμε να αποθηκευτούν τα ευρετήρια μας .

String indexDir="/tmp/testindex";

Με το FSDirectory αποθηκεύουμε τα αποτελέσματα μας.

Directory index = FSDirectory.open(Paths.get(indexDir));

Ορίζουμε το configuration file της Lucene με το οποίο θα γράψουμε

IndexWriterConfig config=new IndexWriterConfig(analyzer);

Δημιουργούμε το configuration file της Lucene με το οποίο θα γράψουμε τα δεδομένα μας σαν ευρετήρια.

IndexWriter w=new IndexWriter (index , config);

Στην συνέχεια με την ίδια διαδικασία όπως και παραπάνω θα διαβάσουμε το αρχείο cacm.all που περιέχει τα κείμενα που θα αξιοποιήσουμε.

FileInputStream fstream = new FileInputStream("cacm/cacm.all");

Διάβασμα Αρχείου cacm.all



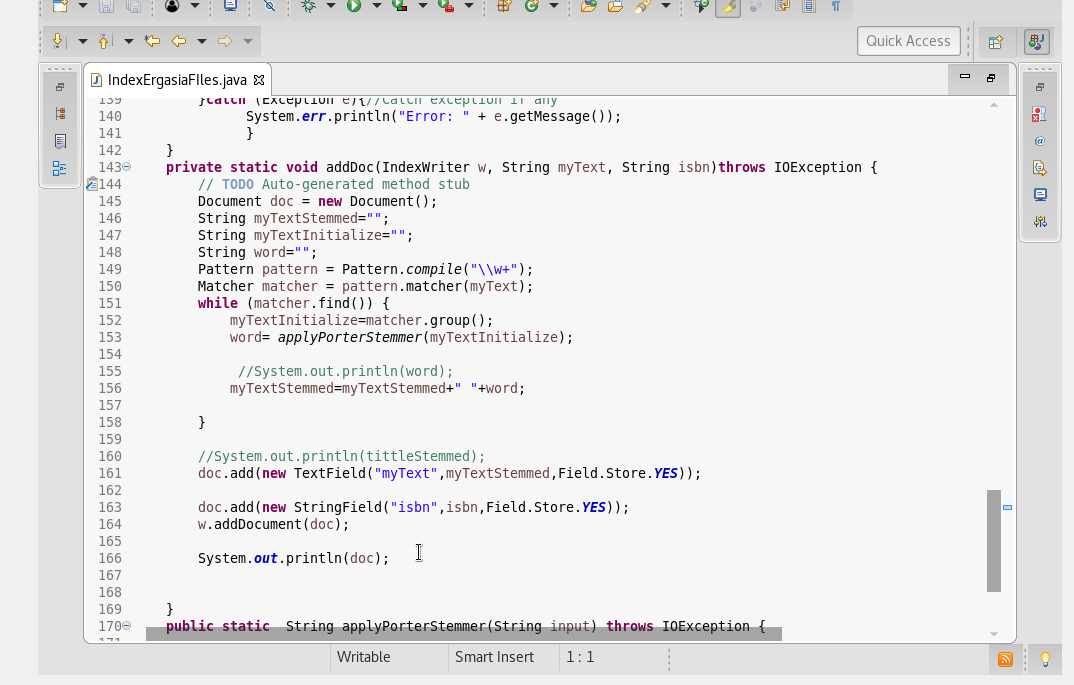


Σύμφωνα με την εκφώνηση θέλουμε να παίρνουμε στα ευρετήρια μας το αρχείο cacm.all τον τίτλο , τον κείμενο και το και τον συγγραφέα . Οπότε καθώς διαβάζουμε το αρχείο όταν θα βρίσκουμε την γραμμή που περιέχει .I και .W και .Α θα το βάζουμε στα ευρετήρια μας.

Οπότε δηλώνων το S2 και το S1 .To S2 παίρνει τον τίτλο ,το περιεχόμενο και τον συγγραφέα κάθε κείμενου .Το S1 παίρνει το αριθμό του κείμενου που θα εισάγουμε στην βιβλιοθήκη Lucene.



Καλώ την μέθοδο με την οποία θα δημιουργήσω τα ευρετήρια με την βοήθεια της Lucene.



Η μέθοδος δέχεται σαν όρισμα την μέθοδο με την οποία θα γράψουμε στα ευρετήρια την οποία ορίσαμε παραπάνω η οποία θα αγνοεί τις λέξεις των κειμένων, παίρνει όλο το κείμενο τίτλο ,κείμενο και συγγραφέα και τέλος παίρνει τον αριθμό του κειμένου που θα περάσει στο ευρετήριο.

**private static void addDoc(IndexWriter w, String myText, String isbn)throws IOException {**

Αρχικά δηλώνουμε το αρχείο που θα δημιουργούμε κάθε φορά.

**Document doc = new Document();**

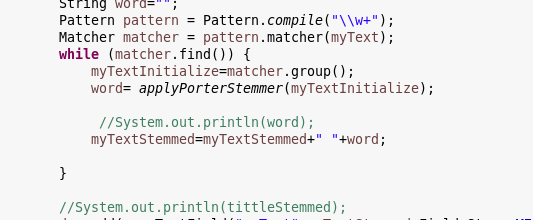
Δηλώνουμε το σχέδιο με το οποίο θα γράφουμε με την βοήθεια της Lucene που πήραμε ως όρισμα και παίρνουμε όλο το περιεχόμενο του κειμένου .

**Pattern pattern = Pattern.compile("\\w+");**

**Matcher matcher = pattern. matcher(myText);**

Η επανάληψη χρησιμοποιείται για να πάρουμε το όλο το περιεχόμενο ανά λέξη και στην ουσία να την κάνουμε tokenize με την Lucene επειδή έχουμε δηλώσει τις λέξεις από το αρχείο common\_words όλες αυτές οι λέξεις που περιέχονται θα αγνοηθούν .Τέλος σε αυτές τις λέξεις όπως μας ζητάει από την εκφώνηση θα εφαρμόσουμε τον αλγόριθμο του Porter για Stemming. Καλώ την ίδια μέθοδο που χρησιμοποίησα κα παραπάνω.

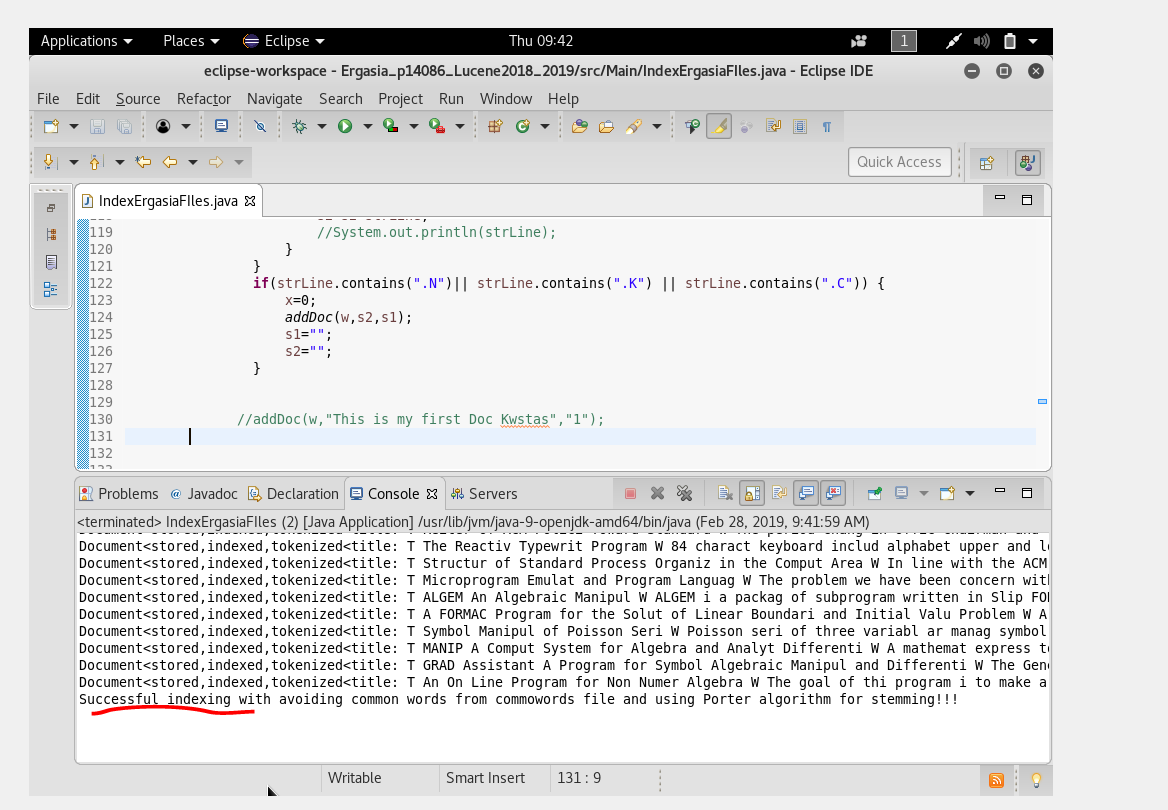
**word= applyPorterStemmer(myText Initialize);**



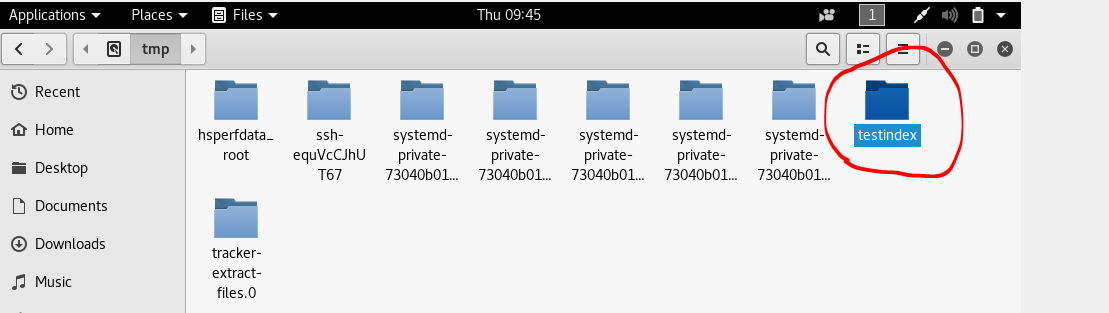
Στην συνέχεια αφού περάσουμε όλες τις λέξεις από τον αλγόριθμο του Porter αποθηκεύουμε το περιεχόμενο μας στα ευρετήρια.

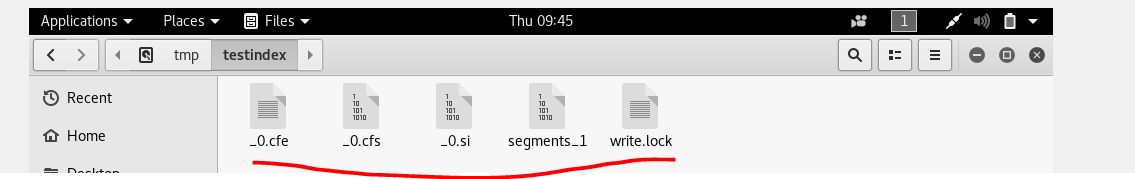


Αν τρέξουμε το αρχείο θα μπορέσουμε να δούμε ότι έχει αποθηκευτεί με ονομασία testindex στο αντίστοιχο path που ορίσαμε tmp/testindex



Όπως βλέπουμε μας εμφανίζει στην κονσόλα πώς αποθηκεύτηκαν τα κείμεναμπορούμε να δούμε ότι έχει χρησιμοποιηθεί πάνω στις λέξεις ο αλγόριθμος Porter για Stemming.



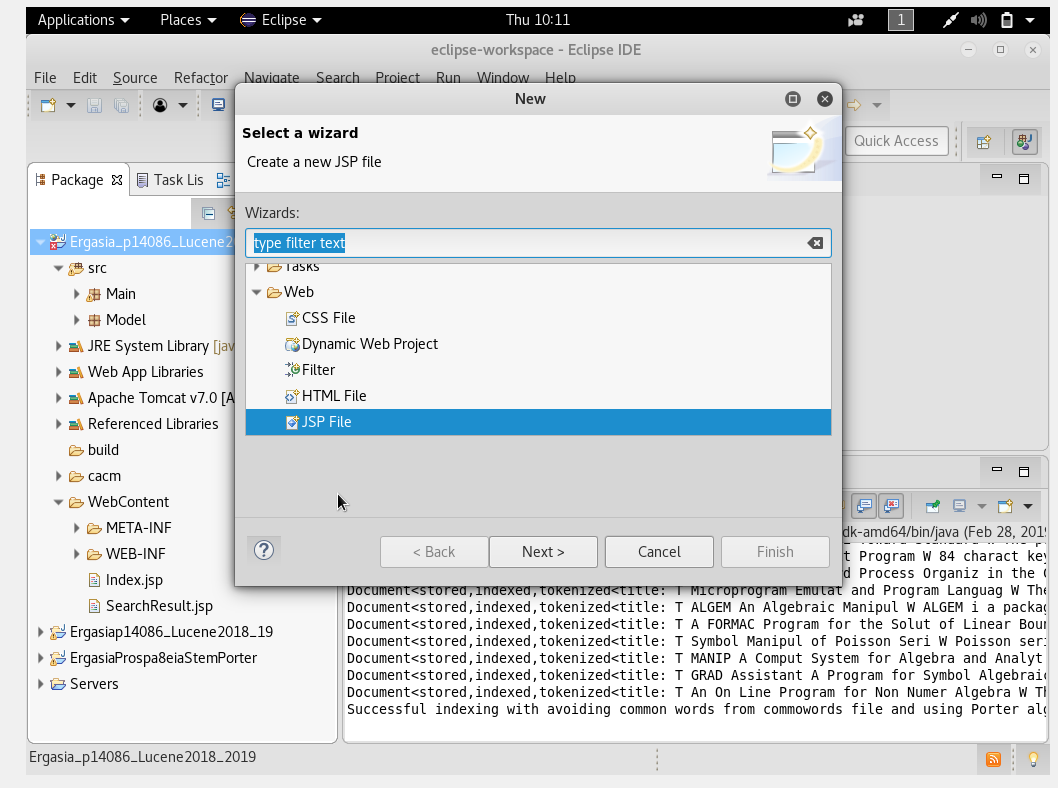


Βλέπουμε και τα ευρετήρια που έχουν δημιουργηθεί .

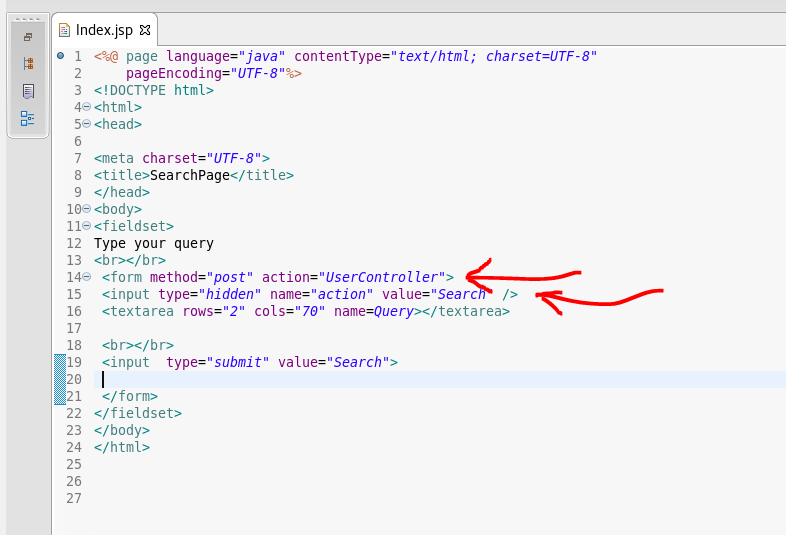
**Δημιουργία Ιστοσελίδας για Ερωτήσεις Πάνω στα Ευρετήρια με την Βοήθεια της Lucene.**

Επιλέγουμε δεξί κλικ πάνω στο Project new ->other ->jsp

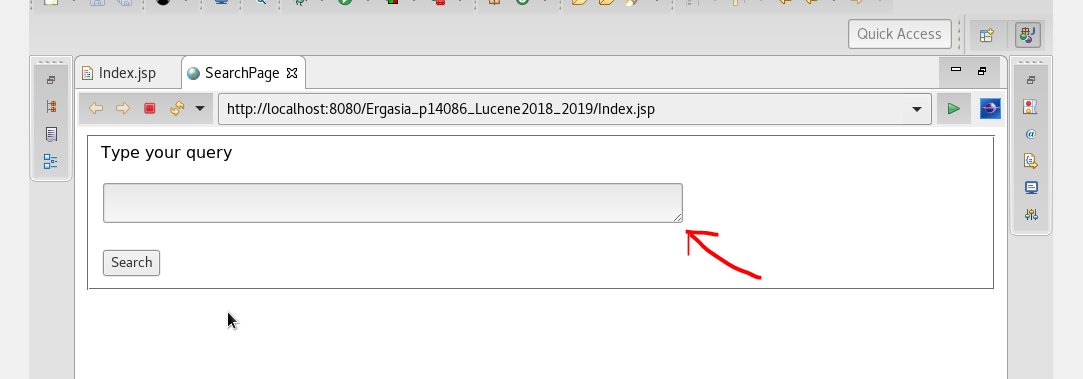
Δημιουργούμε στο web content μια Index.jsp σελίδα



Αυτή η σελίδα είναι η αρχική μας σελίδα που θα πληκτρολογεί ο χρήστης τα ερωτήματα του.



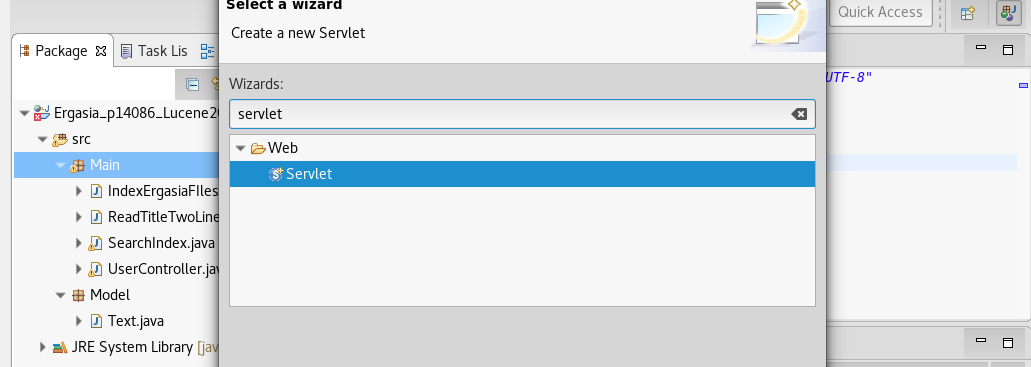
Έχουμε την μέθοδο post με την οποία θα παίρνουμε τα δεδομένα μας και δημιουργούμε ένα action τοοποίο θα το παίρνουμε για να ψάχνουμε το ερώτημα μας στα ευρετήρια τέλος η εντολή <**textarea>** δημιουργεί ένα πλαίσιο το οποίο μπορεί να ρυθμιστεί από τον χρήστη για το μέγεθος για να βλέπει ολόκληρη την ερώτηση με όνομα **query** . Η ονομασία είναι σημαντική γιατί με αυτήν παίρνουμε το περιεχόμενο του πλαισίου για να μπορέσουμε να ψάξουμε στα ευρετήρια.



Αν τρέξουμε την αντίστοιχη ιστοσελίδα μπορούμε να δούμε το διαθέσιμο πλαίσιο και ότι μπορούμε να πληκτρολογήσουμε ερωτήσεις με βάση αυτό. Επιπλέον μπορούμε να ρυθμίσουμε το μέγεθος του πλαισίου.

**Δημιουργία ενός web Servlet**

Δεξί κλικ επιλέγουμε να δημιουργήσουμε ένα Servlet



Με όνομα **UserController.java**



Με την μέθοδο post παίρνουμε με την εντολή request.getParameter τα δεδομένα που έχει πληκτρολογήσει ο χρήστης και το είδος του action Που έχουμε ορίσει.

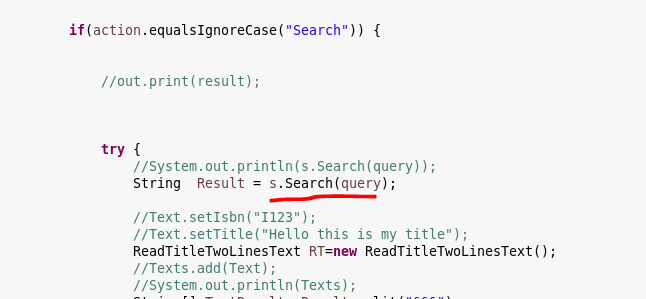
Αφού έχουμε πάρει το ερώτημα του χρήστη δημιουργούμε έναν αντικείμενο για την κλάση Search Index που έχω δημιουργήσει και θα αναλυθεί στην συνέχεια. Παράλληλα ορίζουμε έναν πίνακα λίστα τον οποίο και θα χρησιμοποιήσουμε για να εμφανίζουμε τα αποτελέσματα μας στον χρήστη .



Επιπλέον επειδή θέλουμε σύμφωνα με την εκφώνηση να δημιουργήσουμε για κάθε ερώτημα από το αρχείο που μας έχει δοθεί θέλουμε να φτιάξουμε το διαγράμματα precision-recall θα γράφω τον αριθμό των κείμενων (Isbn) που μας επιστρέφει η Lucene. Οπότε θα δημιουργήσω το αρχείο text1.txt με αυτά τα κείμενα για να τα χρησιμοποιήσω αργότερα.

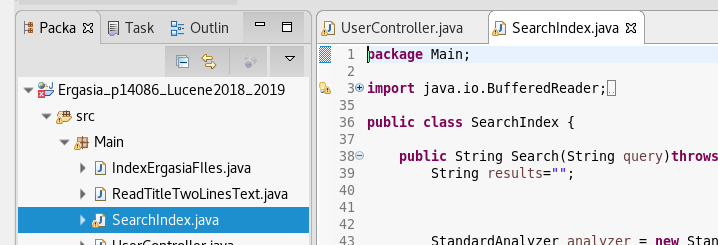
Στην συνέχεια επειδή θέλουμε να γράψουμε text1.txt ορίζουμε αφού δημιουργήσουμε το αρχείο μπορούμε να γράψουμε σε αυτό.

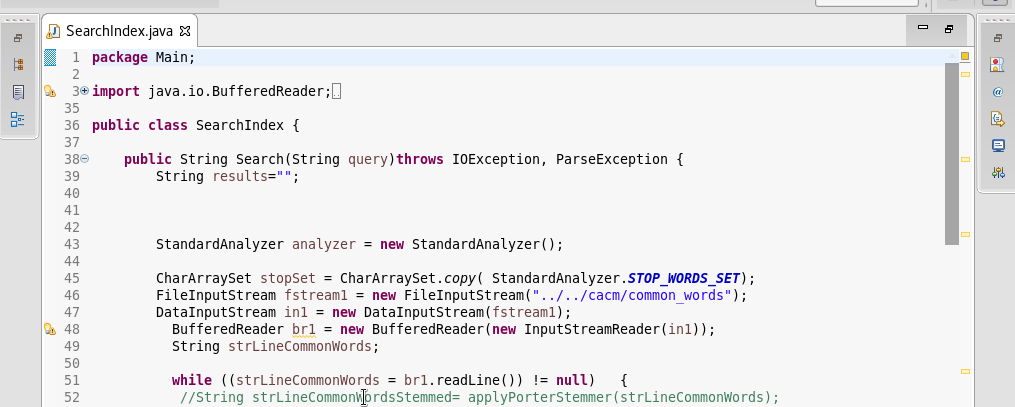
Ελέγχω αν ερχόμαστε από την ιστοσελίδα με action search.



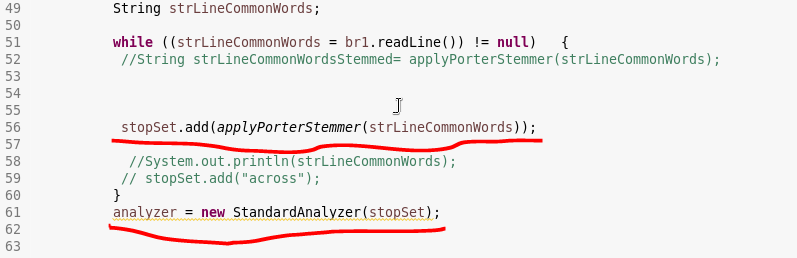
Αφού έχουμε το ερώτημα καλούμε την κλάση Search για να ψάξουμε το ερώτημα στα ευρετήρια που δημιουργήσαμε στην αρχή.

**Ανάλυση Κλάσης SearchIndex.java**





Με την ίδια διαδικασία που δημιουργήσαμε στα αρχείο IndexErgasiaFiles.java δηλαδή θα ορίσουμε τις λέξεις που θέλουμε να αγνοήσουμε από το αρχείο common\_words και θα χρησιμοποιήσουμε στο ερώτημα τον αλγόριθμο του Porter για Stemming.

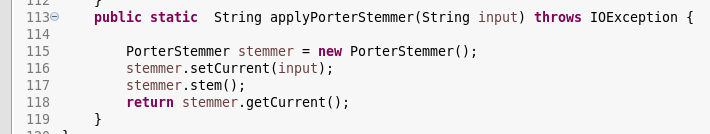


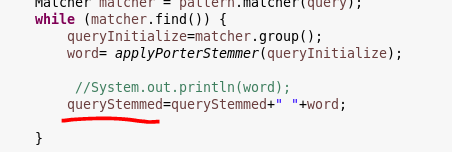
Ορίζουμε με τον ίδιο τρόπο της λέξεις που επιθυμούμε να αγνοήσουμε.



Ορίζω το path που θα πάρω τα ευρετήρια που δημιούργησα και θα τα διαβάσω παίρνω το ερώτημα του χρήστη και το σπάω σε λέξεις για να εφαρμόσω το αλγόριθμο του Porter για Stemming όπως και προηγουμένως .

Έχω δημιουργήσει την μέθοδο applyPorterStemmer που έχω αναλύσει παραπάνω και περνάω το ερώτημα ανά λέξη.





Στο τέλος της επανάληψης θα έχω το ερώτημα σε μορφή του αλγόριθμου Porter.



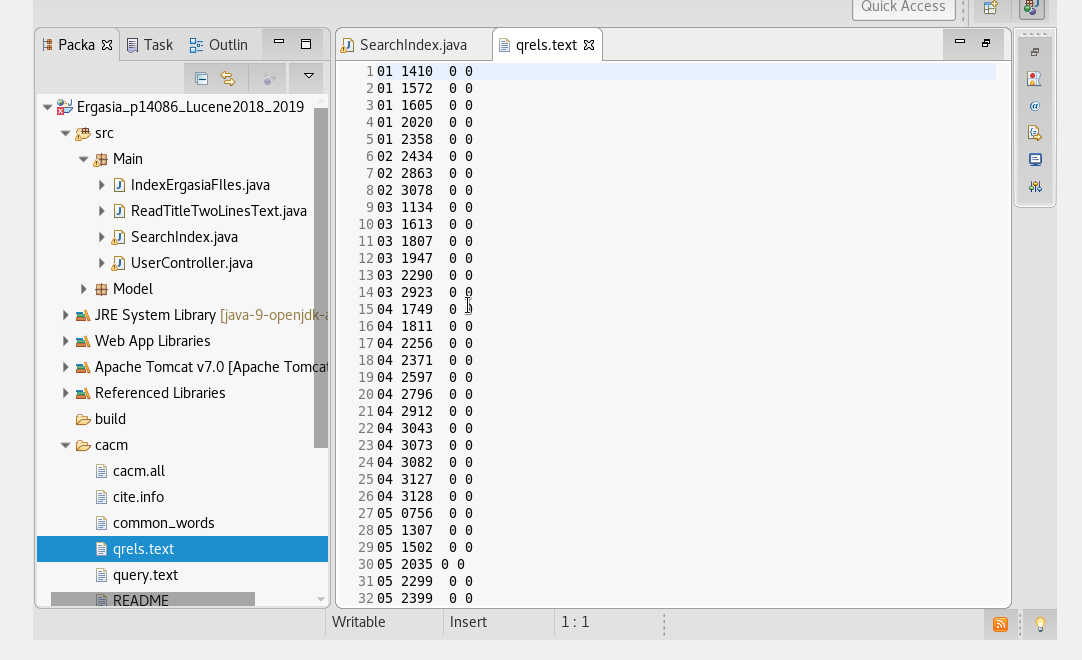
Οπότε ορίζω το configuration για την αναζήτηση με την βοήθεια της Lucene .Ορίζω το νέο ερώτημα του χρήστη με Porter.

Query q=new QueryParser(“myText”,analyzer).parse(querystr)

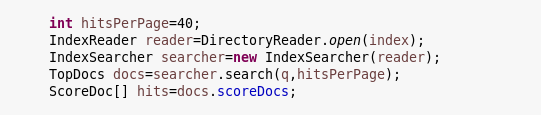
Παίρνω το ερώτημα για να αναζητήσω τα ευρετήρια της Lucene

Int hitsPerPage=40;

Ορίζω πόσα κείμενα θέλουμε να επιστρέφουμε στον χρήστη αν ανοίξουμε το αρχείο qrels.txt που έχει τα σχετικά κείμενα από τα 64 ερωτήματα τα περισσότερα σχετικά που επιστρέφονται είναι 30 κείμενα οπότε μπορώ να ορίσω στο κώδικα μου να μου επιστρέφει 40.



Στην συνέχεια ανοίγω το ευρετήριο και ορίζω ότι θέλω να ψάξω σε αυτά τα κείμενα και να επιστρέψω τα top κείμενα.





Αφού βρω ποια είναι τα κείμενα δημιουργώ μια επανάληψη και παίρνω το Isbn τον αριθμό των κειμένων και αυτούς του αριθμούς των κειμένων τους επιστρέφω καθώς επίσης και το σύνολο αυτών.

**Επιστροφή UserController.java**

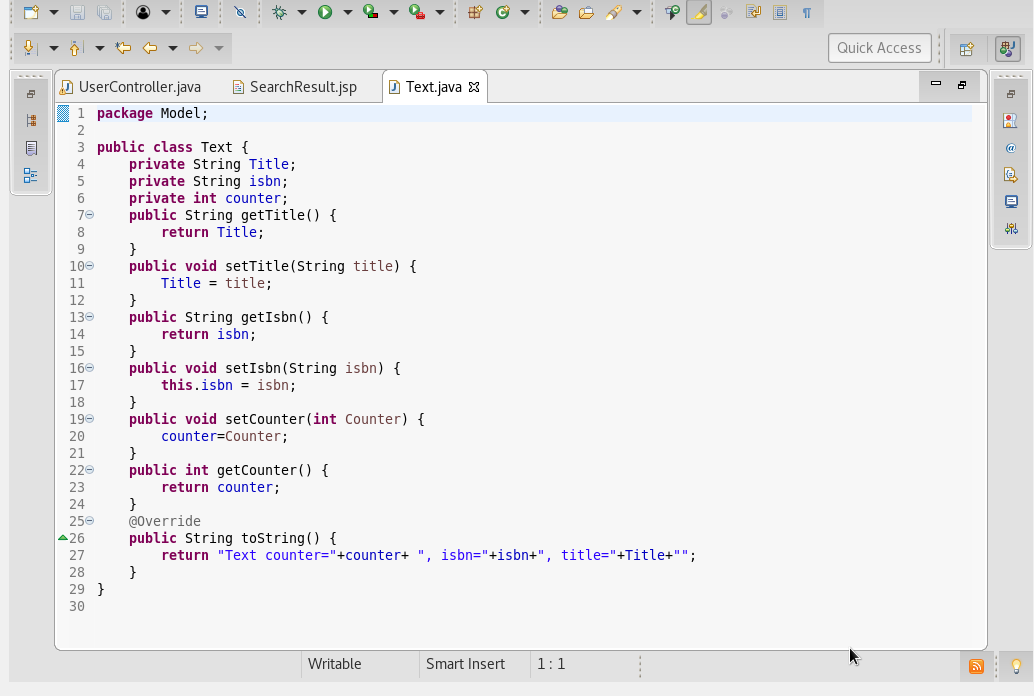
Αφού πλέον έχω το σύνολο των κειμένων και τον αριθμό των κείμενων στην συνέχεια βρισκω από το κανονικό αρχείο cacm.all διαβάζω τον τίτλο και μερικές γραμμές από το κανονικό κείμενο για να εμφανίσει στον χρήστη.

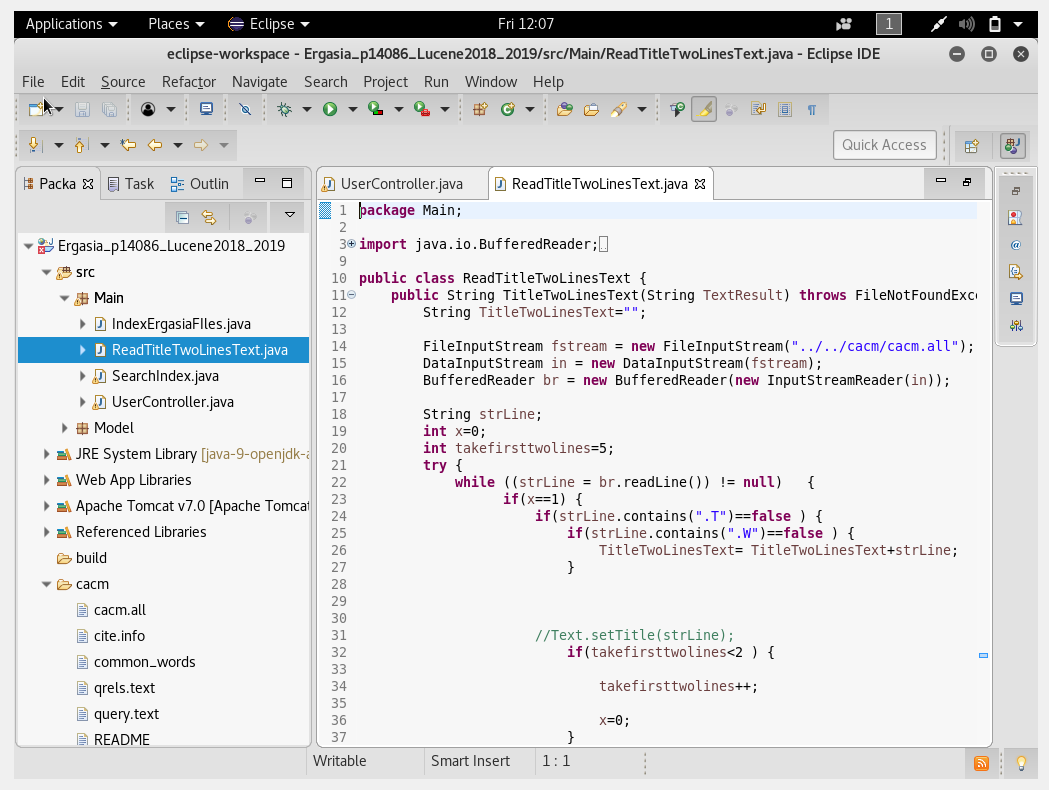


Έχω δημιουργήσει μια κλάση η οποία διαβάζει το κανονικά κείμενα και τα επιστρέφει με τον τρόπο που έχω δείξει και παραπάνω για να διαβάζω αρχεία. Έχω δημιουργήσει την επανάληψη για να βρίσκω κάθε φόρα για κάθε κείμενο το περιεχόμενο ενώ στην συνέχεια κάνω προσθήκη στο πίνακα που ορίσαμε στην αρχή για να τον εμφανίσουμε στον χρήστη στην συνέχεια.

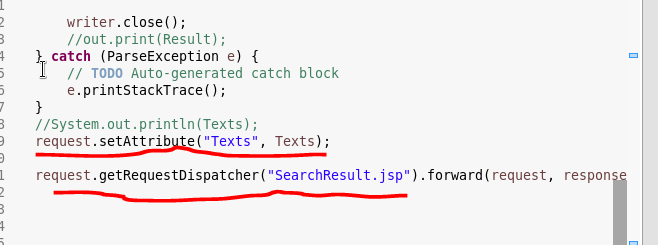
Ενώ παράλληλα γράφω και στο αρχείο text1.txt το κορυφαία κείμενα.

Για να ορίσω τον πίνακα έχω δημιουργήσει μια κλάση Text η οποία δημιουργεί ώστε να βρίσκω την σειρά προτεραιότητας ,τον αριθμό του κειμένου(Isbn),το περιεχόμενο των κειμένων.



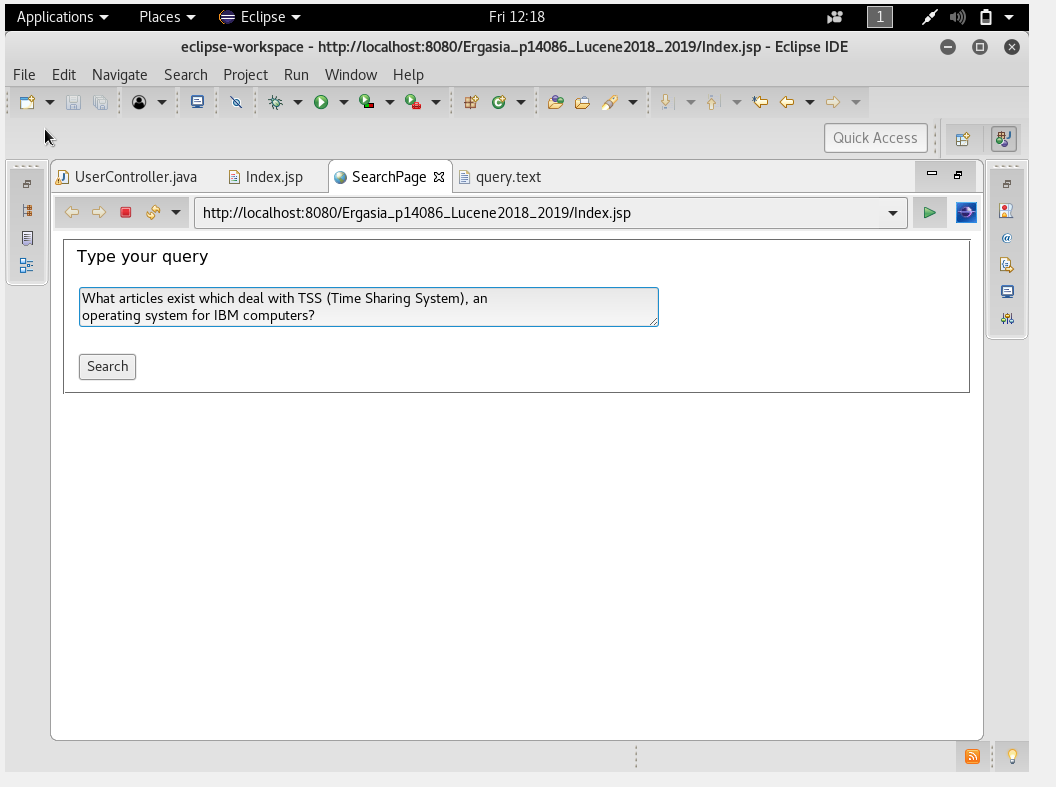


Επιστρέφω το αλφαριθμητικό που περιέχει το περιεχόμενο των τίτλο και τις πρώτες δυο γραμμές κειμένων.

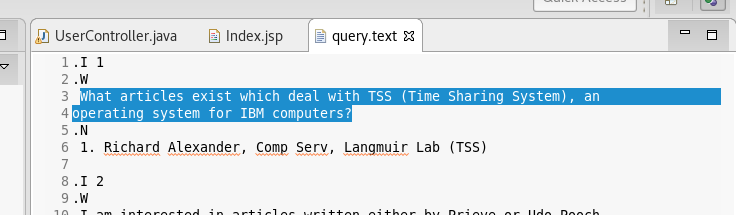


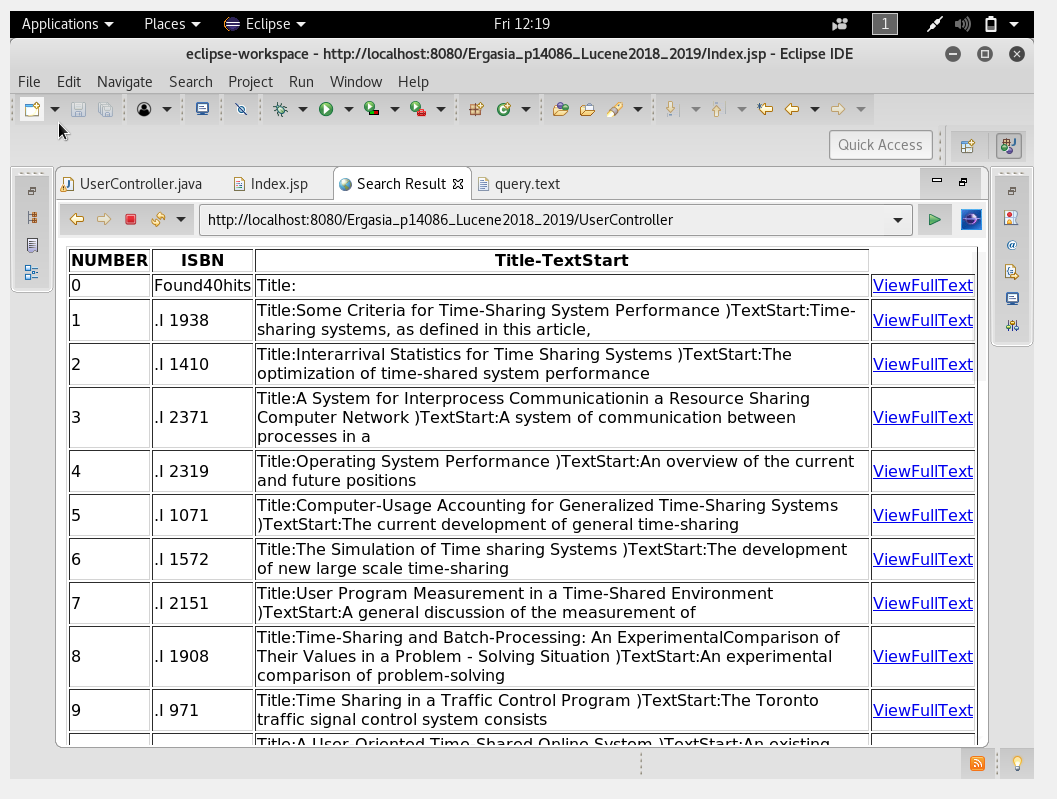
Αφού ορίσουμε το πίνακα με τα διαθέσιμα κείμενα και τους αριθμούς τα παίρναμε με όρισμα Texts για να μπορέσουμε να τα χρησιμοποιήσουμε.

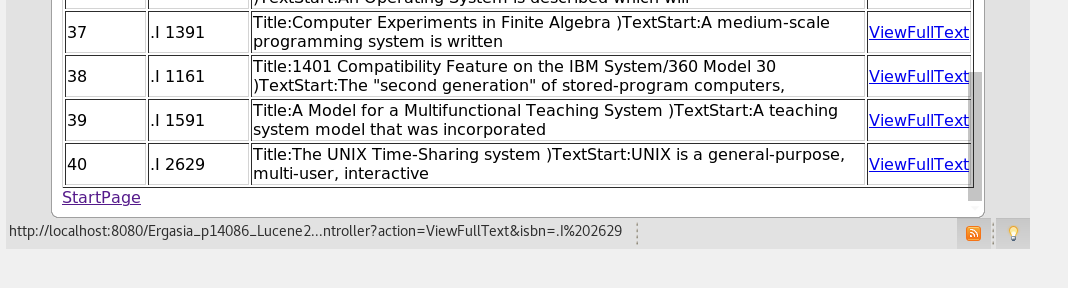
Τέλος προχωράμε στην jsp σελίδα SearchResult.jsp και εμφανίζουμε τα κείμενα και τους τίτλους .



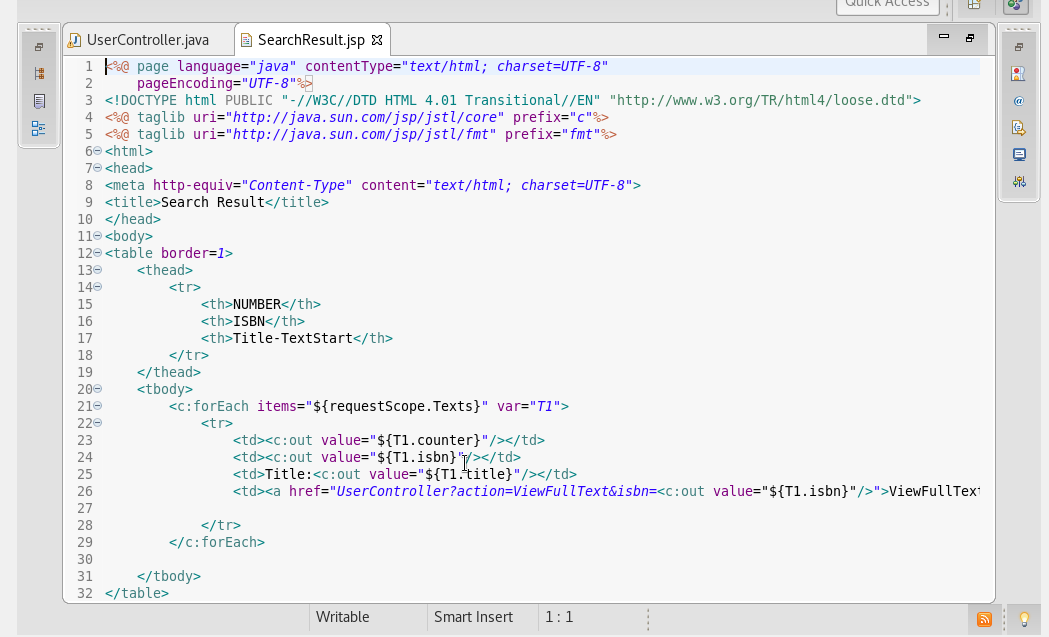
Παίρνω το πρώτο ερώτημα από το αρχείο query.txt για παράδειγμα και κάνω search .







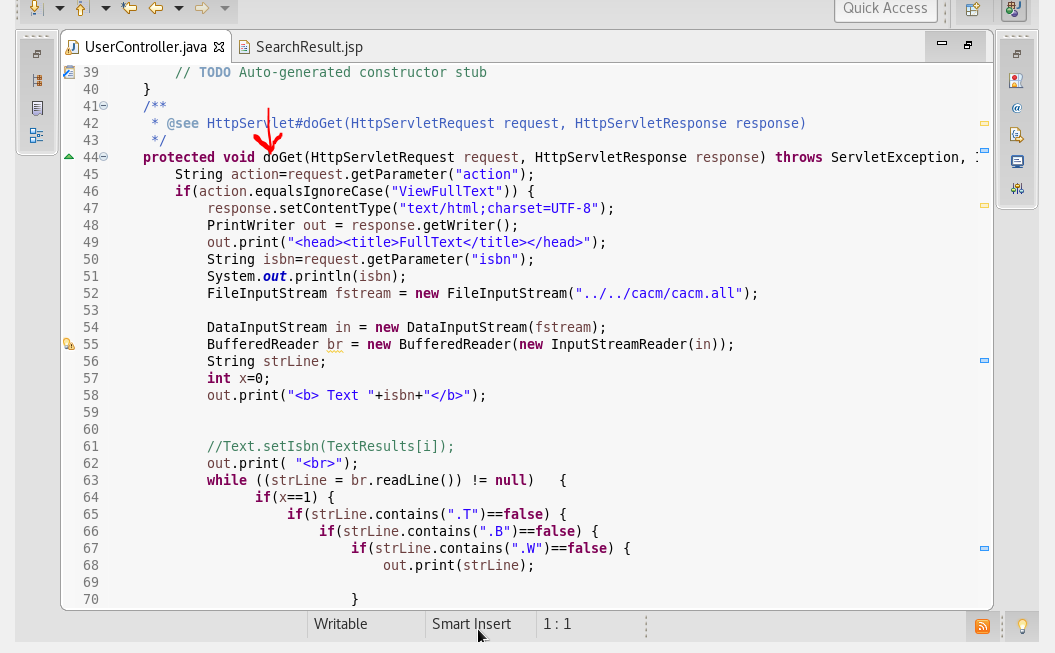
Βλέπουμε ότι μας εμφανίζει στην πρώτη στήλη τον αριθμό κατά σειρά προτεραιότητας στην συνέχεια τον αριθμό του κείμενου Isbn , τον Τίτλο και τις πρώτες γραμμές του κείμενου και τέλος ο χρήστης έχει την επιλογή να διαβάσει ολόκληρο το κείμενο.

Ανάλυση SearchResult.jsp

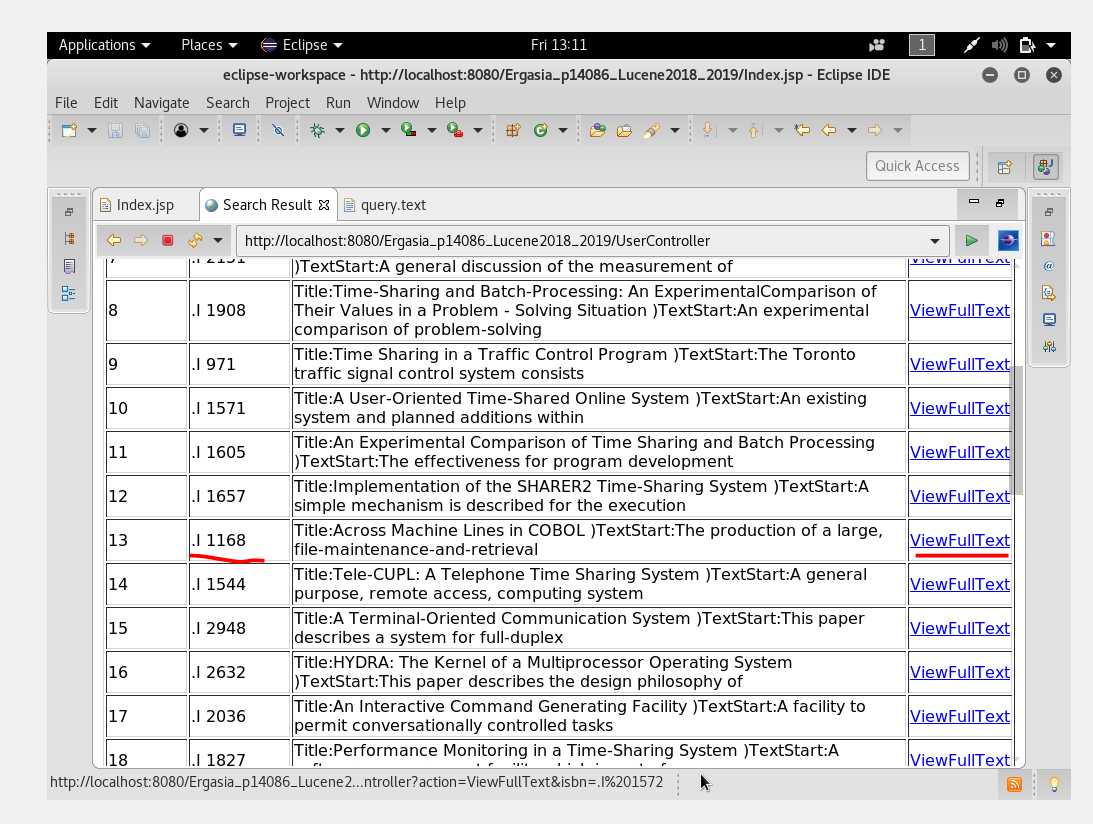
Παίρνω το πίνακα Texts που ορίσαμε παραπάνω και με την εντολή requestScope.Texts παίρνω το περιεχόμενο και με την επανάληψη foreach από εντολές jstl εμφανίζω κάθε φορά το περιεχόμενο.

Για το κουμπί ViewFullText που έχει την επιλογή ο χρήστης παίρνουμε με την μέθοδο get από την δεύτερη στήλη τον αριθμό του κειμένου.

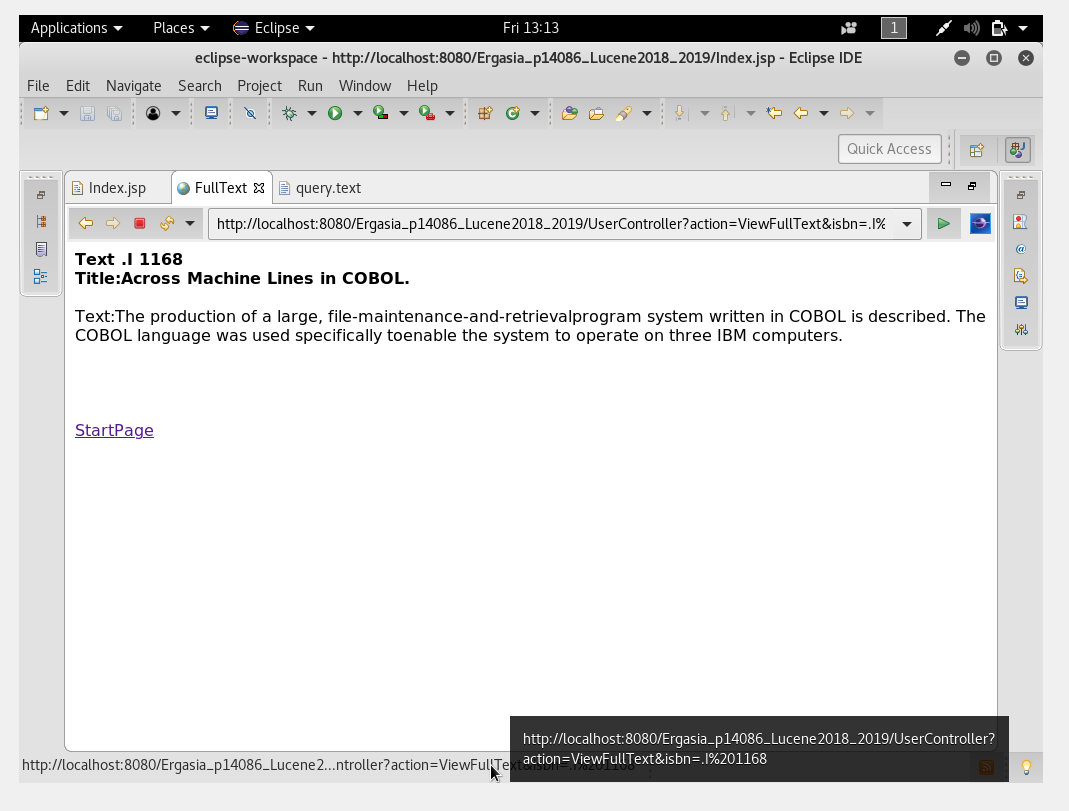
Η εντολή href μας παραπέμπει στο UserController με action ViwFulltext έχοντας σαν μεταβλητή το Isbn.



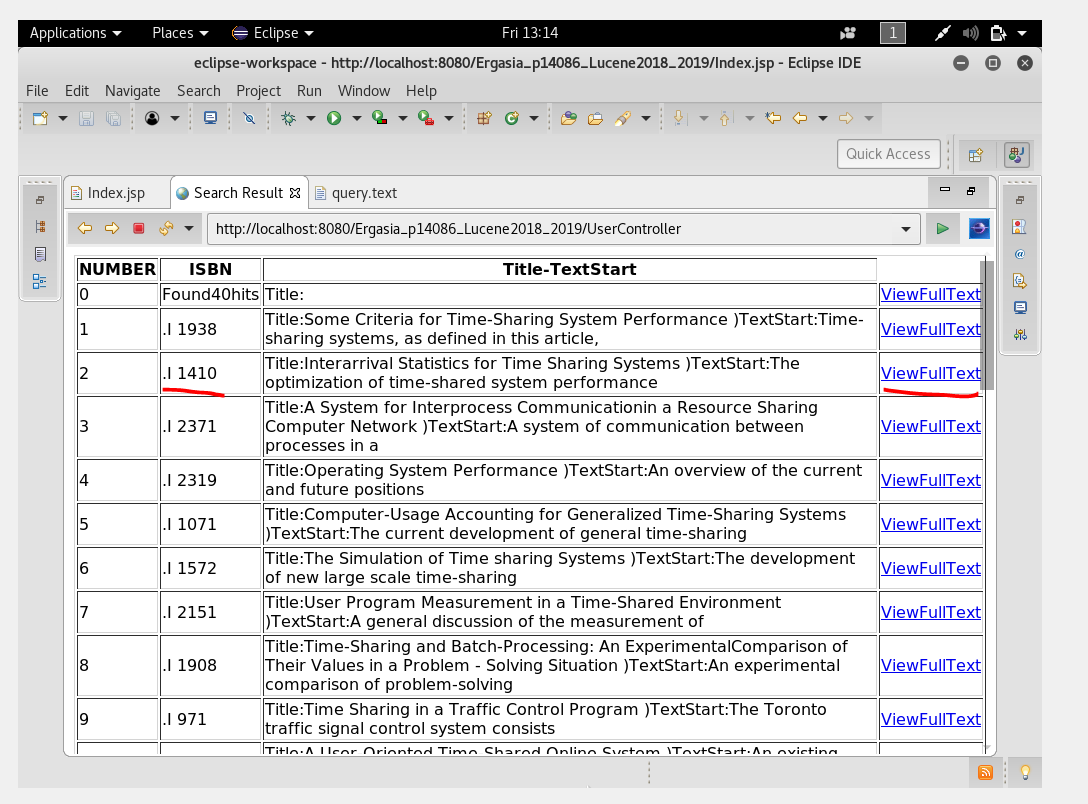
Με την εντολή out.print() γράφουμε στην ιστοσελίδα και με την ίδια διαδικασία διαβάζουμε το cacm.all και εμφανίζουμε ολόκληρο το κείμενο μας.

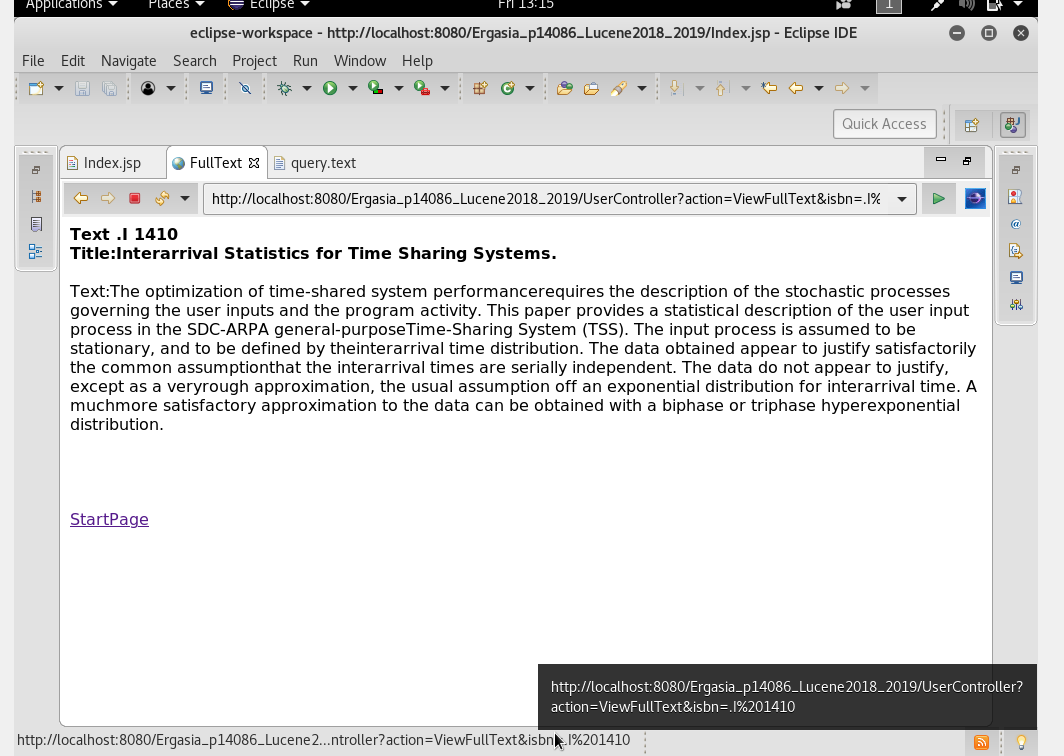


Έστω ότι διαβάζουμε τυχαία το κείμενο .Ι 1168 επιλέγοντας ViewFullText



Και μας εμφανίζει το Isbn τον τίτλο και το κείμενο αναλυτικα.





Αντίστοιχα για το .Ι 1410.

**Πηγές:**

<https://www.baeldung.com/lucene-analyzers>

<http://makble.com/how-to-add-custom-stop-words-in-lucene>

<https://stackoverflow.com/questions/18008999/how-to-add-custom-stop-words-using-lucene-in-java>

το χρησιμοποίησα για να βάλλω τις λέξεις από το αρχείο common\_words και να τις αγνοώ όταν θα εμφανίζονται στο σύστημα

<https://github.com/caarmen/porter-stemmer>

<https://stackoverflow.com/questions/15422485/lucene-porter-stemmer-not-public>

το χρησιμοποίησα για τον αλγόριθμο του Porter για Stemming