

İSTANBUL TEKNİK ÜNİVERSİTESİ

Bilgisayar ve Bilişim Fakültesi

Powerful Search in the Forum

STAJ

Abdullah AYDEĞER

YAZ / 2012

İstanbul Teknik Üniversitesi
Bilgisayar ve Bilişim Fakültesi
STAJ RAPORU

Akademik Yıl: 2011-2012

Staj yapılan dönem: Yaz Bahar Güz

Öğrenci ile ilgili bilgiler

Adı ve Soyadı: Abdullah AYDEĞER

Öğrenci Numarası: 040090533

Bölüm: Bilgisayar Mühendisliği

Program: Bilgisayar Mühendisliği

e-posta adresi: aydeger@itu.edu.tr

(Cep) Tel No: 0538 448 99 08

ÇAP öğrencisi misiniz?: Evet (ÇAP yaptığınız Fakülte/Bölüm: _____)

Hayır

Mezuniyet Evet

durumunda mısınız: Hayır

Yaz okulunda ders Evet (Ders sayısı: __)

alıyor musunuz? Hayır

Öğrencinin çalıştığı kurum ile ilgili bilgiler

İsmi: Data Storage Institute

Birim: Data Center Technologies

Web <http://www.dsi.a-star.edu.sg/Pages/index.aspx>

Adres:

Kısa DSI Building, 5 Engineering Drive 1,
adres: (Off Kent Ridge Crescent, NUS),
Singapore 117608
Republic of Singapore

Yetkili kiři ile ilgili bilgiler

Bölümü:

Ünvanı: Arařtırmacı(Research Scientist)

Adı ve Soyadı: Yongqing ZHU

(kurumsal) e-posta: zhu_yongqing@dsi.a-star.edu.sg

(kurumsal) Tel No:

Yapılan iş ile ilgili bilgiler

Staj yeri Türkiye

Yurtdışı

Staj başlangıç tarihi 18.06.2012

Staj bitiş tarihi 14.09.2012

Stajda çalışılan net **gün** sayısı 60

Staj süresinde sigortanız var mıydı? Evet, İTÜ tarafından sigortalandım

Evet, kurum tarafından sigortalandım

Hayır, yurtdışı stajı yaptım

Hayır

Information of the Project

In these days, by improved technology, softcopy documents are more preferred than hardcopies for most of the companies and some of these companies has immense sources on the computers. This creates lots of information on the computers. Therefore, finding the real, which is really wanted, and necessary word is getting hard. For handling this problem there is one research topic in the computer engineering which can be called “Information Retrieval”. This topic especially focuses on both developing efficient and accurate techniques.

Information Retrieval has some sub-topics. One of this is web search. In my project I investigated web search techniques. After that, I was occupied to implement a search engine in the forum website. I was supposed to implement firstly a forum website with some features such as post addition, deletion and update and also categorizing the whole forum topics-posts. After writing the forum website I added search engine in the forum website. While engine is run, requested word are searched in the database and sorting the found results according to the ranking formula, in the forum website. For sorting of found results, post’s length, found frequency and post’s level (is that post first entry for that title or second or more) have been considered.

Before mentioning about technical details of the project, some features about search engines should be explained.

Search Engines

Search engine can be described as the software program that enables users to search word(s) and show found results on the screen. But this work cannot be done so simple. There are many algorithms to searching relevant words. These algorithms should be based on not only running time but also accuracy of the search results. They all have a purpose for finding relevant ones fastly. By defining relevant, one good example can be given as: if any user searches for ‘cheap university stuff’, ‘inexpensive book’ may be related to the query and should be showed to the user.

What should be considered?

While implementing search engines following terms should be considered.

Efficiency of Indexes: Time of creating indexes should be logical, must not take too many times.

Efficiency of Queries: Time of querying some terms should not take times which users may be bored to wait.

Effectiveness of Queries: Queries should find most of the related terms to user input.

Scalability: Search engine should work for very large files too. It should not be limited any size of data.

How should be implemented?

It should be implemented with following components:

Index Builder: Search engines should not waste time for scanning all documents for each query. Instead of that, inverted index should be created while saving operations are processing. Inverted index means indexing each meaningful in the entries (for our forum website, entry can be called post) by using it's post information (such as post length, post level), it's position in the post. For being clearer to inverted index, following example is useful.

Example post (id: 1): I have a problem with my database.

Example post (id: 2): Can you express your problem with details? In addition, database management is really hard thing.

Inverted indexes: (have, 1, 2) , (problem, {(1, 4), (2, 5)}), (with, {(1, 5), (2, 6)}), (database, {(1,6), (2, 10)}), (express, 2, 3), (your, 2, 4), (details, 2, 7), (addition, 2, 9), (management, 2, 11), (really, 2, 13), (hard, 2, 15), (thing, 2, 16)

In above record, first term shows exact word, second one is id of the post where that word is included, third one is word's position in that post. Only some words are considered which are longer than 3 characters.

But most of the search engines do not save the exact word in the inverted index while index builder is processing. Instead, it uses stemming algorithms to get rid of some suffixes and making the words stemmed. In addition some small words are removed too. For example of these:

The example entry 1: "computers" will be saved as comput because it's root is that.

The example entry 2: "any of the birds" will be saved as bird (only the word bird will be saved, others will discard because of the length is too small to search).

Furthermore some search engines do not index some words which can include some swears or are common. These words are called stopwords and should be defined as static in the programming environment.

In index builder there should be also tokenizer to split entry into the some blanks or punctuations. For an example if user enters I.E.T.T how it should be saved as an inverted index is quite important problem for search engines. (2 choices are like saving it as I.E.T.T or split it as I and E and T and T)

Query Processing

Every search engines should have query processor with some features of searching such as only one word search, phrase search or detailed search. This query processor should find relevant entries.

Researchers worked for doing good query processors. Good query processors means finding relevant terms fully.

For an example; if inexpensive houses are searched, cheap homes also should be found and showed to the user. For doing better query processors some techniques are designed. First one is using thesaurus while searching into database. But it does not meet the requirements entirely and it is not effective to search a lot of words into large files. Second technique is relevance feedback. This uses feedback mechanism to search for better. Even though there are some methods, it is still not very possible to find all related terms in the search engines.

How my search engine is implemented?

I followed below steps to implement my search engine in the forum website.

- **Parsing the search entry:** Parsing can be divided as 2 steps.
 - ✓ Separate each word from the entry between each punctuation signs.
 - ✓ Make each word stemmed according to grammar rules.

- **Query with the stemmed words from the index table:** Stemmed words should be searched from database.

- **Show the results on the screen:** Found results should be showed on the screen according to their rankings (that differs from one algorithm to any other).

Before searching any word, some words have already been in the database. While saving data to database, parsing operation needs to be implemented before saving.

Development and Running Environments

I worked on Fedora for this whole project and I implemented forum website in NetBeans IDE. I created one C and one Java (Web application) project from NetBeans. I followed similar steps as <http://netbeans.org/kb/docs/cnd/beginning-jni-linux.html> website for creating Java+C integrated project. For integration of Java code with C code, I changed some properties of the projects. In addition, I used MongoDB database for saving data permanently. I used MongoDB not only from Java code but also from C code and for using it I also added some commands to NetBeans properties. In below all details about these are explained by screen photos.

For MongoDB, you must run the executable mongod file with root permission. It was in the Home/Download/mongodb-linux-i686-2.0.7-rc1/bin/ folder in my working computer. If you want to test some queries you can run mongo executable, after running mongod, and you can try some queries on that terminal. MongoDB can be installed from <http://www.mongodb.org/downloads>.

For Java project, it has no difference than creating normal Java Web Application project as seen in figure 1. But inside the Java class, which will call C code, this code should be added:

```
static {
    System.load("full-path-to-NetBeansProjects-
dir/JNIDemoCdl/dist/libJNIDemoCdl.so"); //JNIDemoCdl is C file name
}
```

Creating the Native Library Header File

For creating header file which is necessary for C code, listed steps should be followed:

- In a terminal window, navigate to the NetBeansProjects directory.
- Type the following:

```
javah -o JNIDemoJava.h -classpath JNIDemoJava/build/classes jnidemojava.Main
```


Then C header file (JNIDemoJava.h) is generated in the NetBeansProjects directory. This header is required to provide a correct function declaration for the native method(s). It will be needed later when the C part of this application is created.

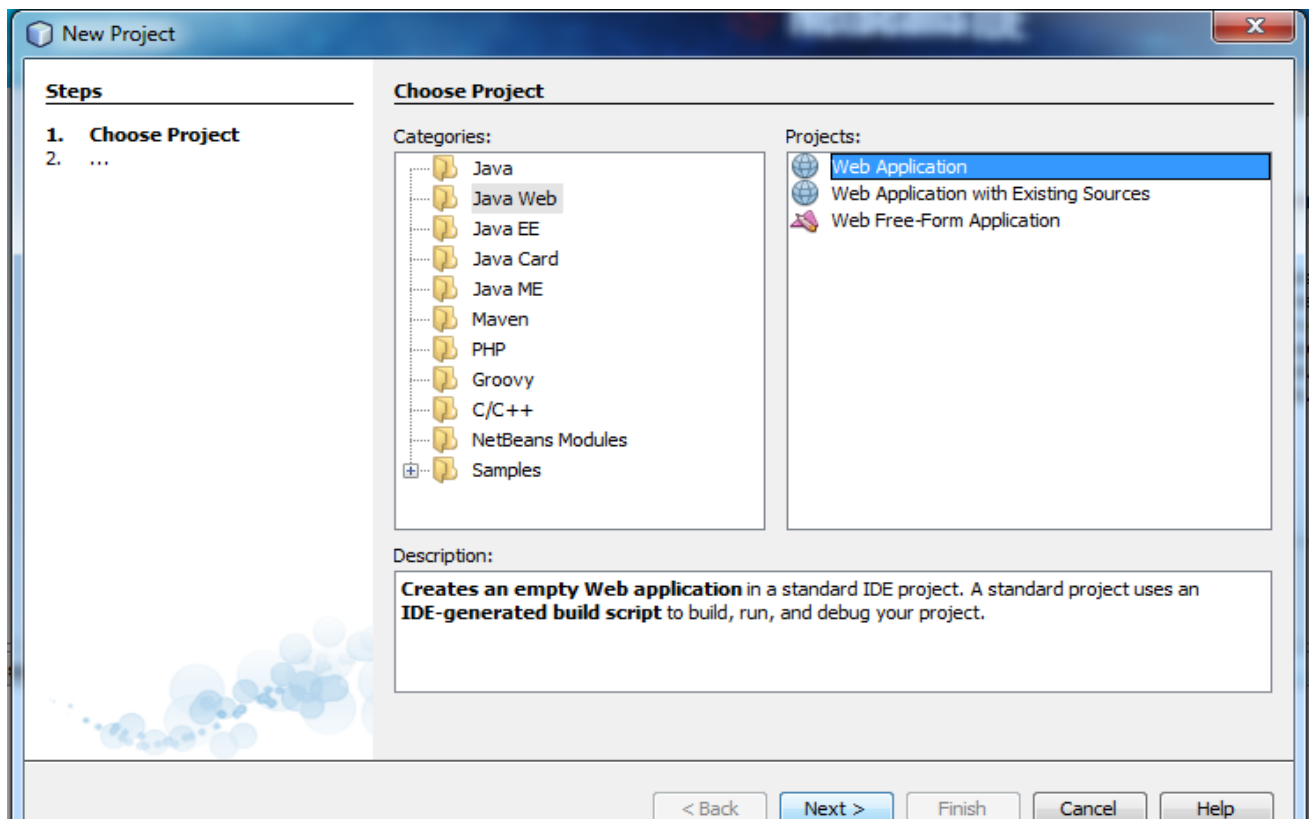


Figure 1. Creating Java Web App

For C Project, C/C++ Dynamic Library project should be created as normally (as seen in figure 2). After creating this project, some properties must be changed to be able to calling from Java code and also to be able to work with MongoDB. These property changes like:

- Choose Build – C Compiler. Change ‘Include Directories’ and browse and add JDK directory and JDK’s include/linux directory (for linux) also add MongoDB libraries folder (as seen in figure 4).
- Choose Build – C Compiler. Change ‘Additional Options’ `-shared -m32` for JNI (Java Native Interface) and also add `-std=99` for MongoDB (as seen in figure 5).

- Choose Build – Linker. Change ‘Output’ text with dist/libFORUM.so where libFORUM is C source code name (as seen in figure 6).
- After all of these steps, you can call C code, which has been already compiled, from any Java code.

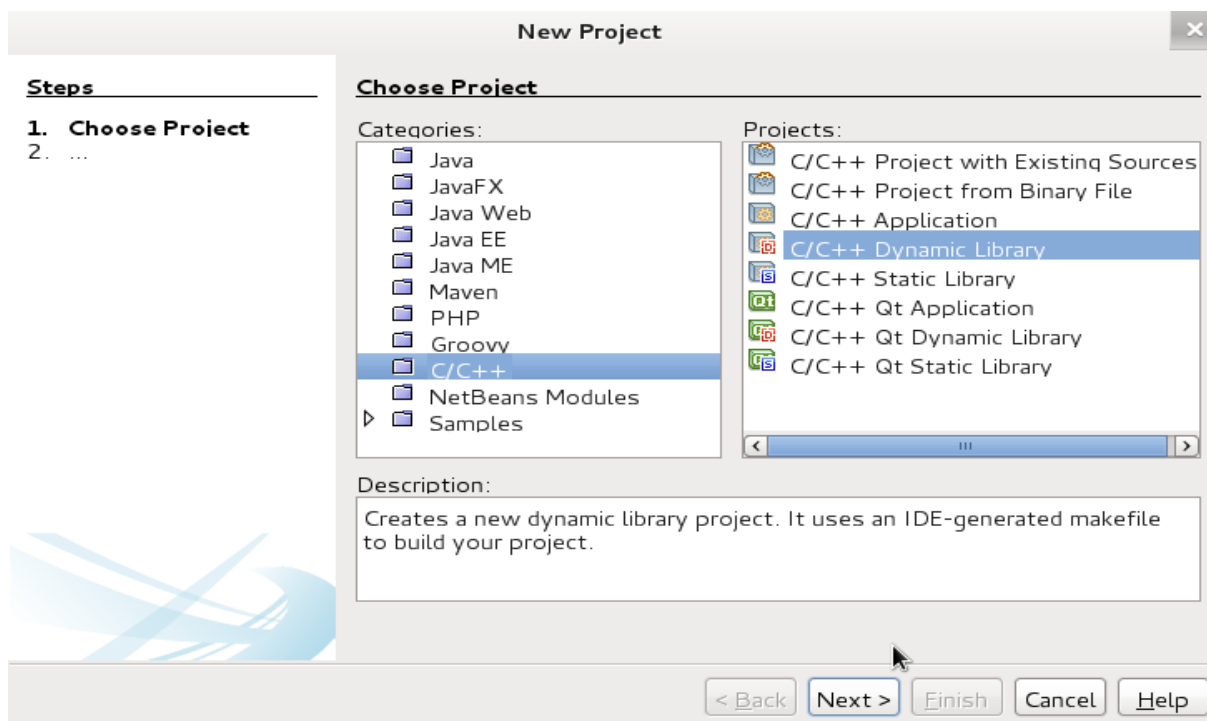


Figure 2. Creating C Project

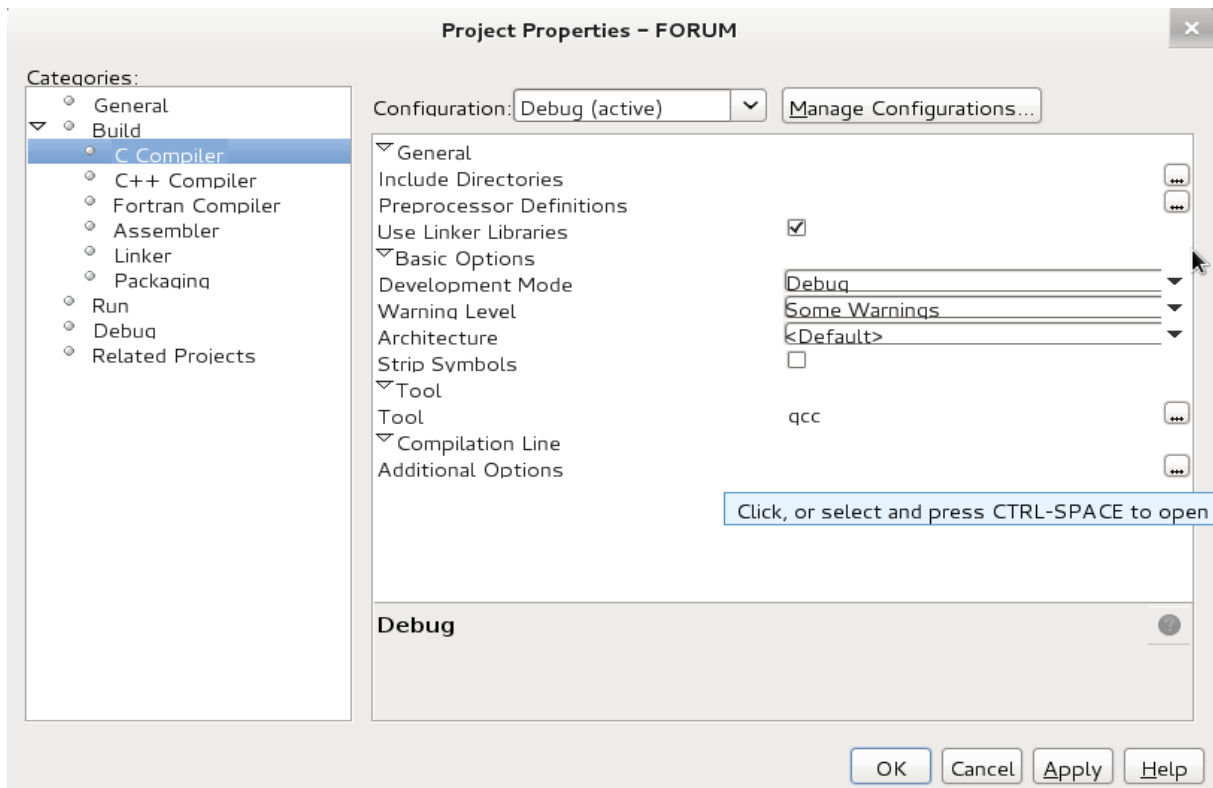


Figure 3. Properties of C Project

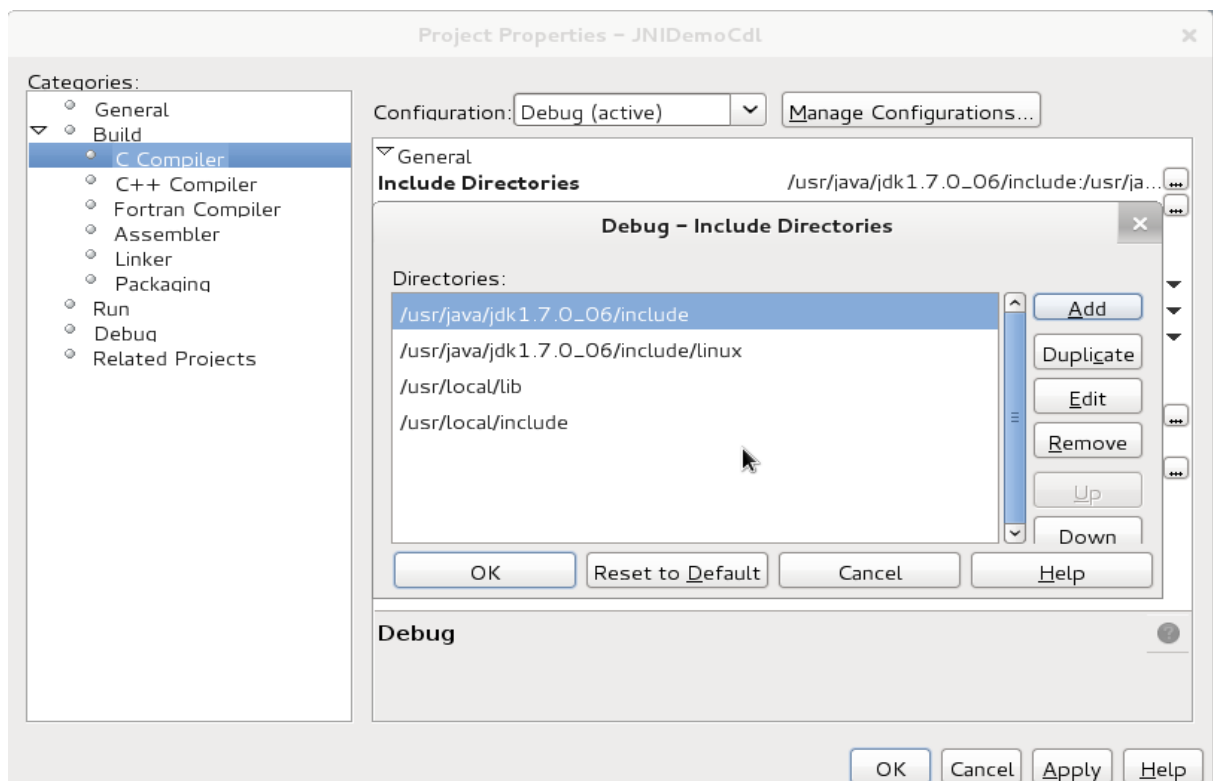


Figure 4. Changing include directories

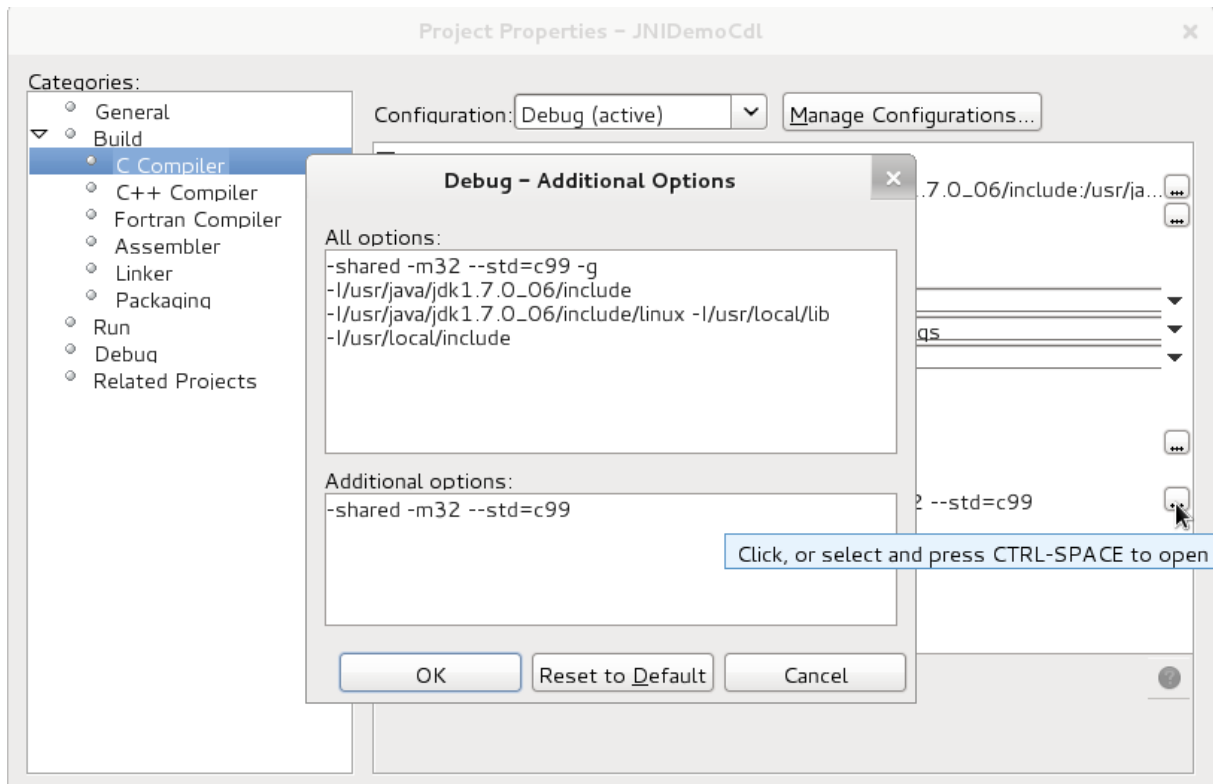


Figure 5. Changing additional options

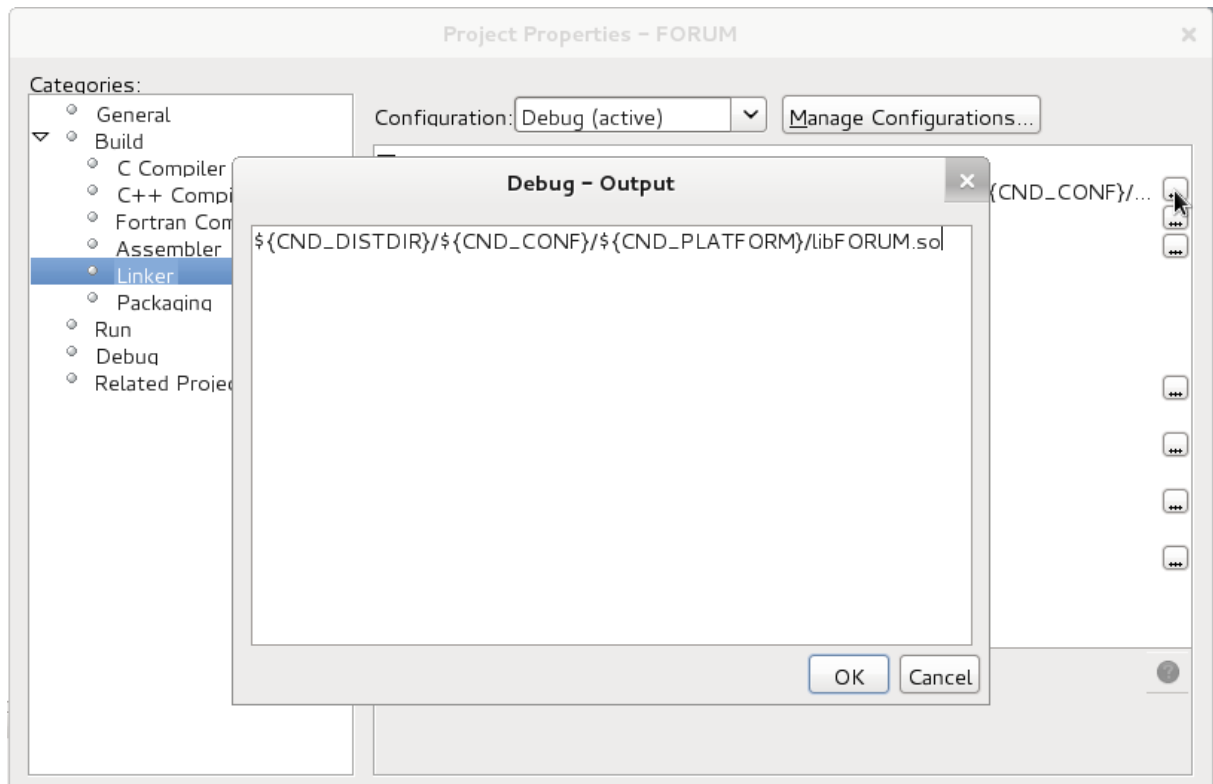


Figure 6. Changing Linker - output

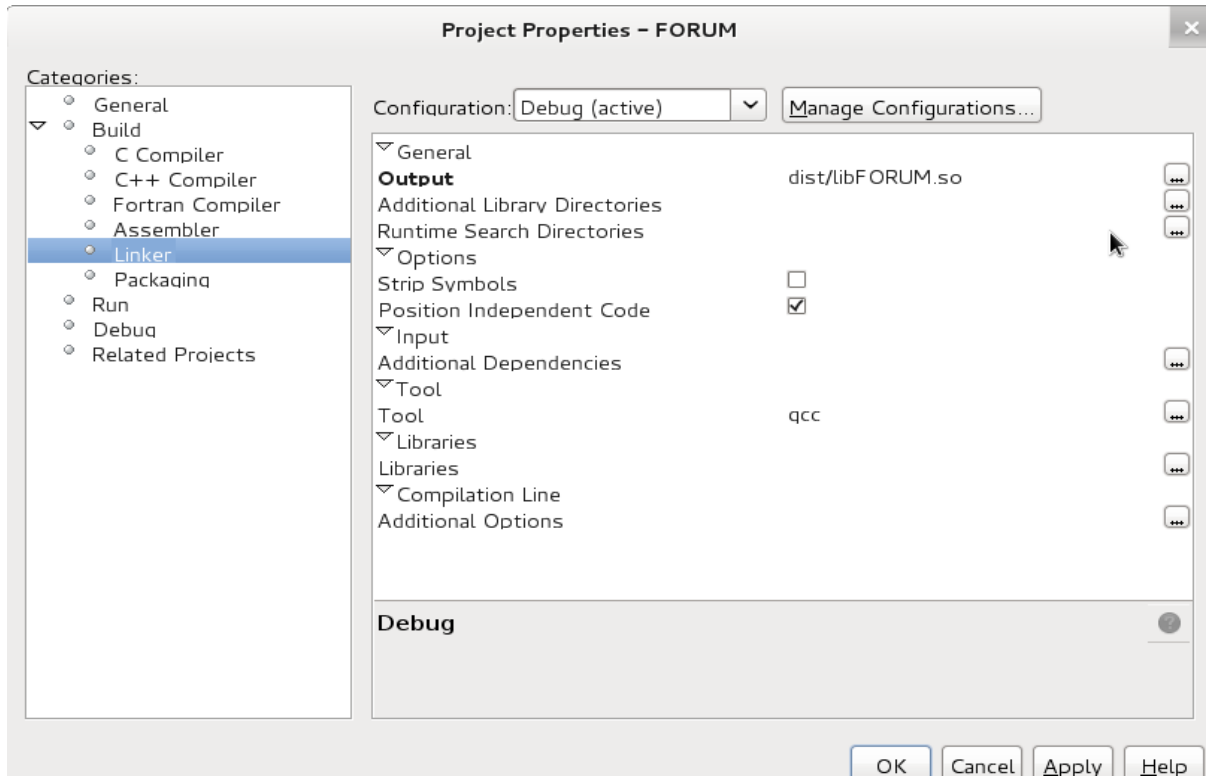


Figure 7. After changing output

Project Codes

I wrote forum website with Java language in JSP Framework. But because of the speed of the running time, I integrated Java code with C code for some parts such as stemming.

Language C Parts

C Header (JNIDemoJava.h)

```
JNIEXPORT jstring JNICALL Java_jnidemojava_JNIDemoJava_stem
    (JNIEnv *, jobject, jstring, jint, jint, jint );

JNIEXPORT jstring JNICALL Java_jnidemojava_JNIDemoJava_deleteIndex
    (JNIEnv *, jobject, jint, jstring);

JNIEXPORT jstring JNICALL Java_jnidemojava_JNIDemoJava_search
    (JNIEnv *, jobject, jstring);

JNIEXPORT jstring JNICALL Java_jnidemojava_JNIDemoJava_phraseSearch
    (JNIEnv *, jobject, jstring);
```

Figure 8. C Functions

C functions' tasks are explained in the source code with details.

```
typedef struct {
    int postID;
    int level;
    int length;
    int tf;
    int titleID;
    int size;
    double searchFeature;
    int *positions;
    int numOfPositions;
    char **actualWords;
} indexTableData;
```

Figure 9. C Data Structure

This structure is used for holding the post data by entirely in one variable.

C Source Code(JNIDemo.c)

Porter Stemmer

I decided to use porter stemmer for stemming the words and I searched for open source Porter Stemmer. I found 'www.cs.cmu.edu/~callan/Teaching/porter.c' website for downloading the source code and with some changes I adapted this code to my project.

```
/* This is the Porter stemming algorithm, coded up in ANSI C by the
author. It may be regarded as cononical, in that it follows the
algorithm presented in

Porter, 1980, An algorithm for suffix stripping, Program, Vol. 14,
no. 3, pp 130-137,

only differing from it at the points made --DEPARTURE-- below.

The algorithm as described in the paper could be exactly replicated
by adjusting the points of DEPARTURE, but this is barely necessary,
because (a) the points of DEPARTURE are definitely improvements, and
(b) no encoding of the Porter stemmer I have seen is anything like
as exact as this version, even with the points of DEPARTURE!

You can compile it on Unix with 'gcc -O3 -o stem stem.c' after which
'stem' takes a list of inputs and sends the stemmed equivalent to
stdout.

The algorithm as encoded here is particularly fast.
*/
```

Figure 10. Porter Stemmer copyrights

Stem function

```
JNIEXPORT jstring JNICALL Java_jnidemojava_JNIDemoJava_stem
(JNIEnv *env, jobject obj, jstring p, jint postID, jint titleID, jint level)
] {
] /*
-   This function make stemmed giving all text and inserting those
-   in the database table with name indexTable
-   */
```

Figure 11. C function with name stem

DeleteIndex function

```
JNIEXPORT jstring JNICALL Java_jnidemojava_JNIDemoJava_deleteIndex
(JNIEnv *env, jobject obj, jint pid, jstring p){
    /*
     * This function deletes terms from indexTable which has postID='pid'
     */
}
```

Figure 12. Delete function

Search function

```
JNIEXPORT jstring JNICALL Java_jnidemojava_JNIDemoJava_search
(JNIEnv *env, jobject obj, jstring p){
    /*
     * This function search the given string(one word) from the indexTable
     * and return back to postID + titleID in string format
     * (without using ranking formula)
     */
}
```

Figure 13. Search function

PhraseSearch function

```
JNIEXPORT jstring JNICALL Java_jnidemojava_JNIDemoJava_phraseSearch
(JNIEnv *env, jobject obj, jstring p){
    /*
     * This function search the given string(one word) from the indexTable
     * and return back to postID + titleID + positions(for phrase searching) in string format
     * (without using ranking formula)
     */
}
```

Figure 14. Phrase search function

Language Java Parts

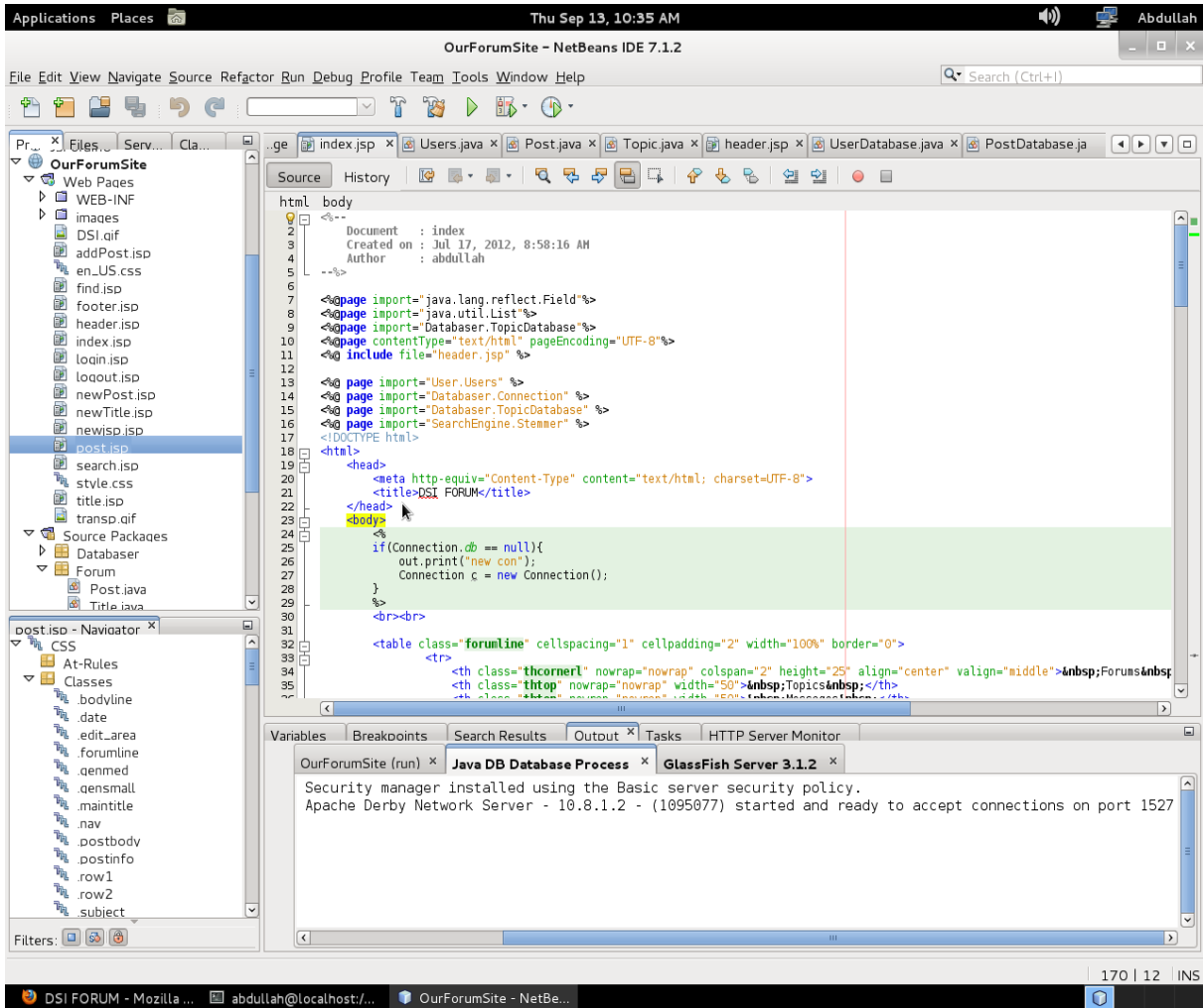


Figure 15. Web page files of forum website(at the left)

```
private native String stem(String p, int a, int b, int c); //for inserting
public native String deleteIndex(int p, String t); //for deleting
public static native String search(String p); //for normal searching
public static native String phraseSearch(String p); //for phrase searching
```

Figure 16. Java native(will call C code) functions

Forum Website Pages

The screenshot displays the NetBeans IDE interface for a project named "OurForumSite". The main editor window shows the source code of "index.jsp". The code defines a table structure for a forum listing. It includes a header row with columns for "Topics", "Messages", and "Last Message". Below the header, there is a row for a forum topic titled "SCIENCE". The topic row contains a folder icon, the topic name "MATH", and a link to the topic page. The code also includes a row for the number of titles for the topic "math", which is displayed as "1".

```
html body table tr td
31 :able class="forumLine" cellspacing="1" cellpadding="2" width="100%" border="0">
32 <tr>
33 <th class="thcornerl" nowrap="nowrap" colspan="2" height="25" align="center" valign="middle">&nbsp;&nbsp;&nbsp;Foru
34 <th class="thtop" nowrap="nowrap" width="50%">&nbsp;&nbsp;&nbsp;Topics&nbsp;&nbsp;&nbsp;</th>
35 <th class="thtop" nowrap="nowrap" width="50%">&nbsp;&nbsp;&nbsp;Messages&nbsp;&nbsp;&nbsp;</th>
36 <th class="thcornerr" nowrap="nowrap">&nbsp;&nbsp;&nbsp;Last Message&nbsp;&nbsp;&nbsp;</th>
37 </tr>
38
39
40 <!-- START FORUM LISTING -->
41 <tr>
42 <td class="catleft" colspan="2" height="28"><span class="cattitle">SCIENCE</span></td>
43 <td class="catright" align="right" colspan="3">&nbsp;&nbsp;&nbsp;</td>
44 </tr>
45
46 <tr>
47 <td class="row1" valign="middle" align="center" height="50">
48 
49 </td>
50 <td class="row1" width="100%" height="50">
51 <span class="forumLink"><a class="forumLink" href="title.jsp?topic=math">MATH</a></span><br />
52 <span class="genmed">
53 This is a test forum
54 </span>
55 <br />
56 </td>
57 <td class="row2" valign="middle" align="center" height="50">
58 <span class="gensmall"><%= TopicDatabase.getCountOfTitles("math") %></span></td>
59 <td class="row2" valign="middle" align="center" height="50">
```

The IDE interface includes a Project Explorer on the left showing the file structure, a Navigator window below it showing the DOM tree, and a Variables/Outout window at the bottom right showing the output of the application run.

Figure 17. Index page source

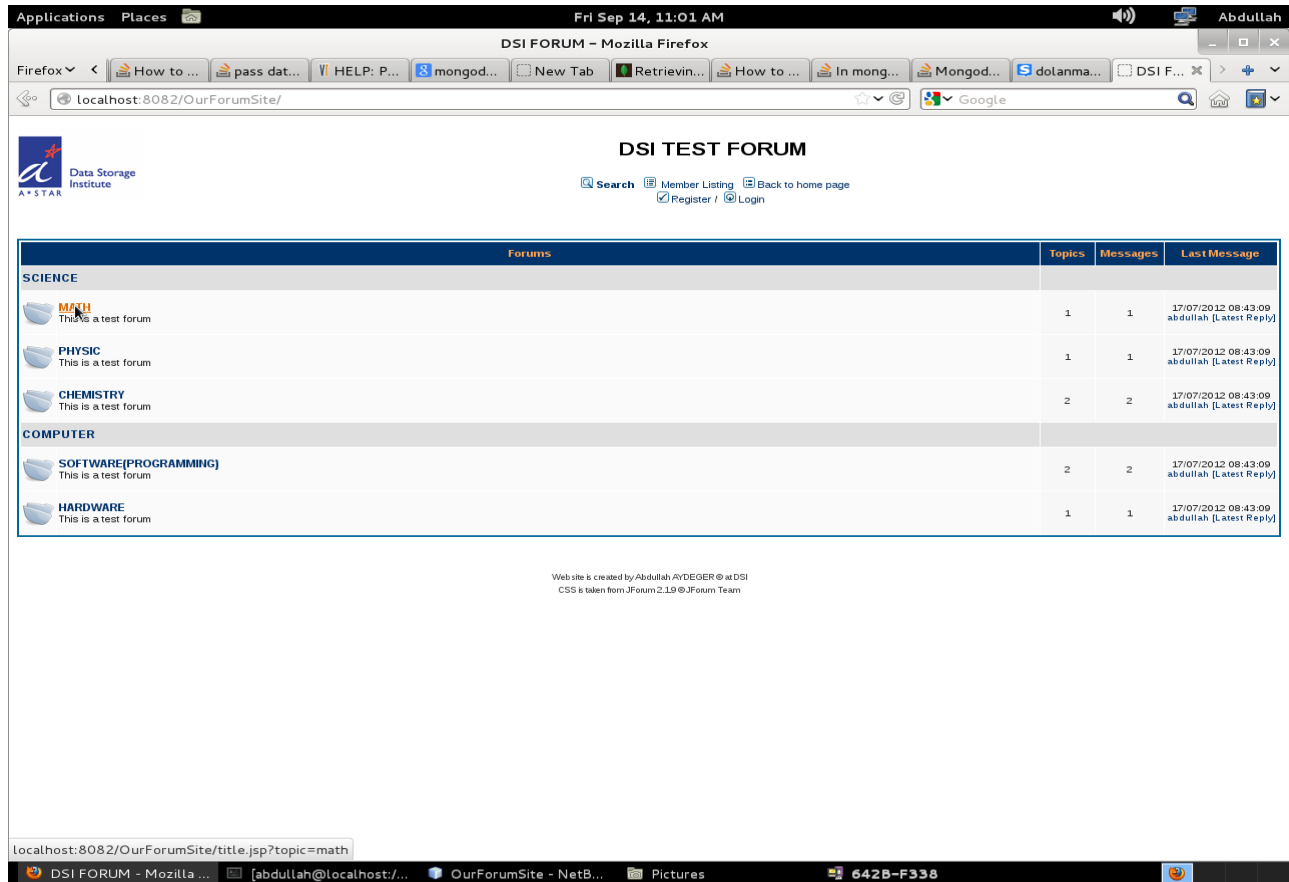


Figure 18. Index Page, topic choosing

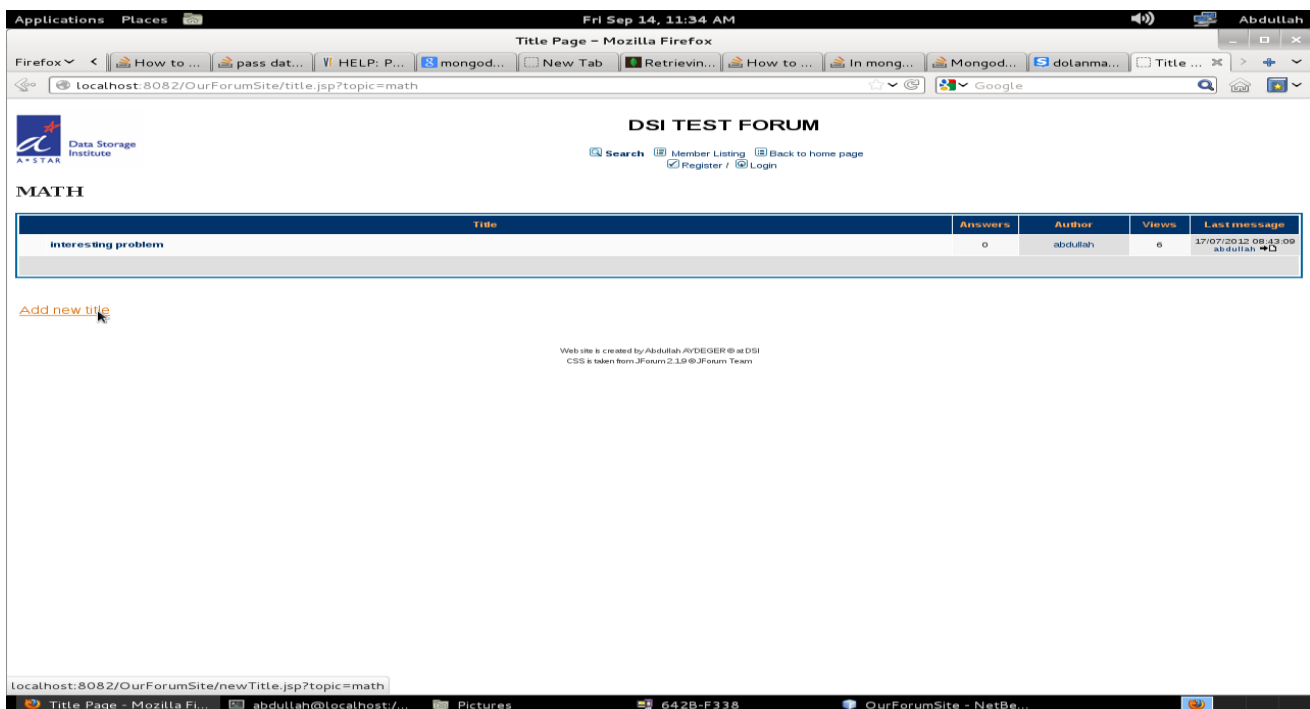


Figure 19. Title page, link of new title addition page

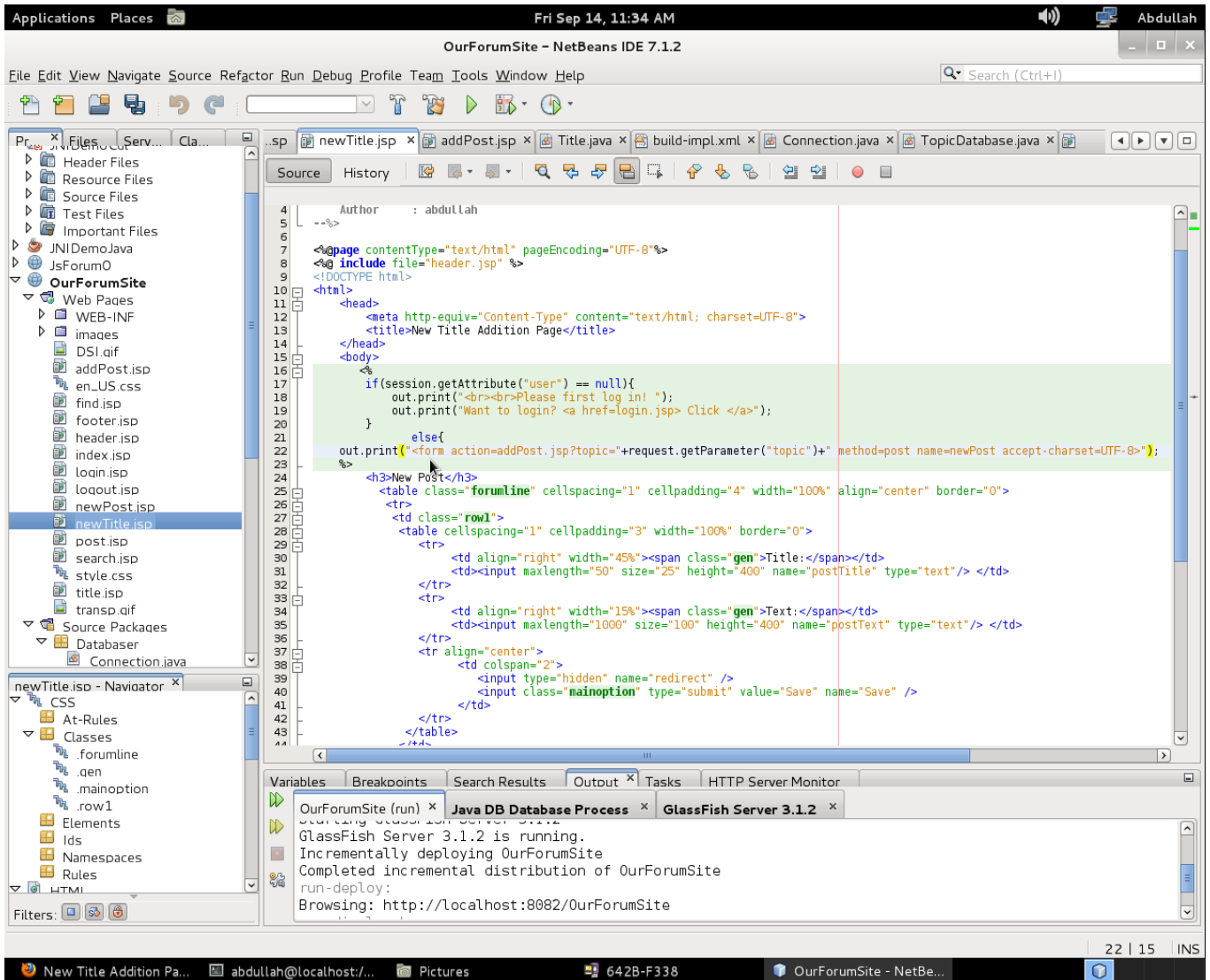


Figure 20. New title addition page source code

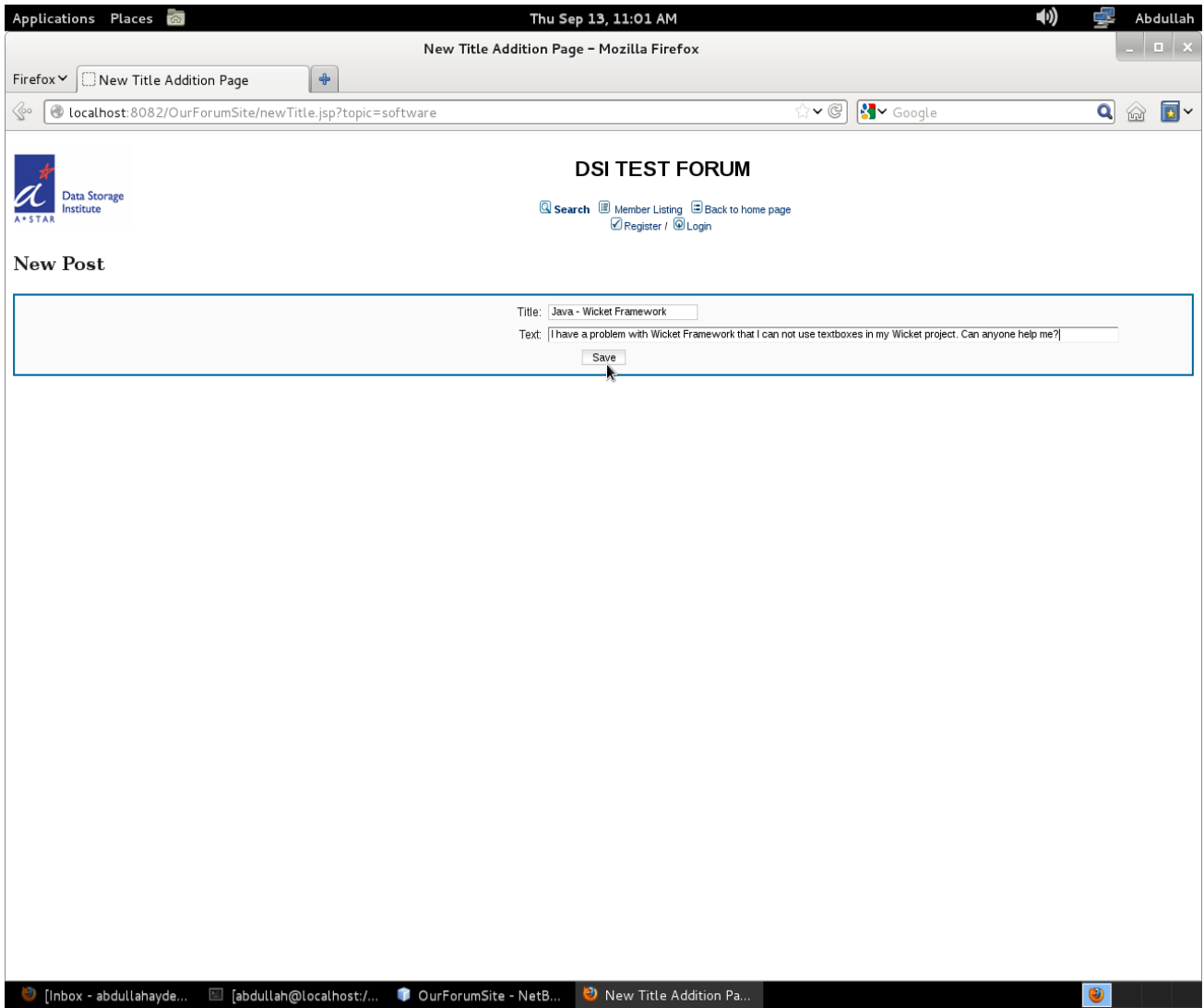


Figure 21. New title addition page

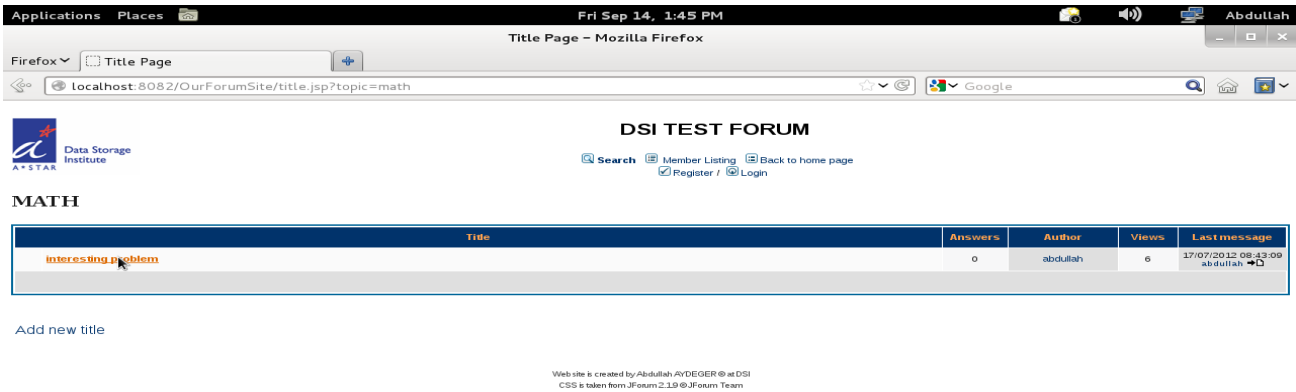


Figure 22. Title page, title choosing

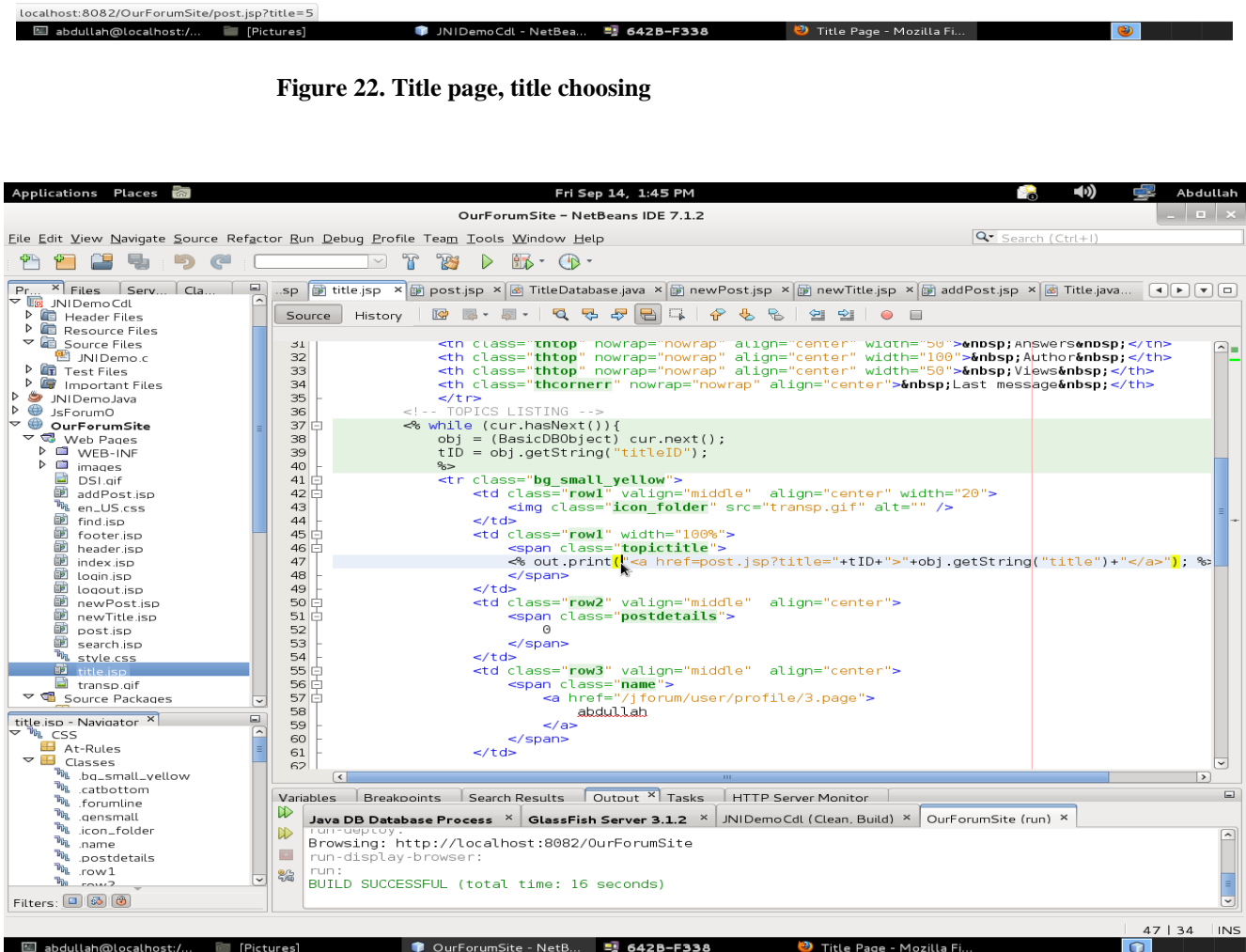


Figure 23. Source code of link(for posts) in the title pages

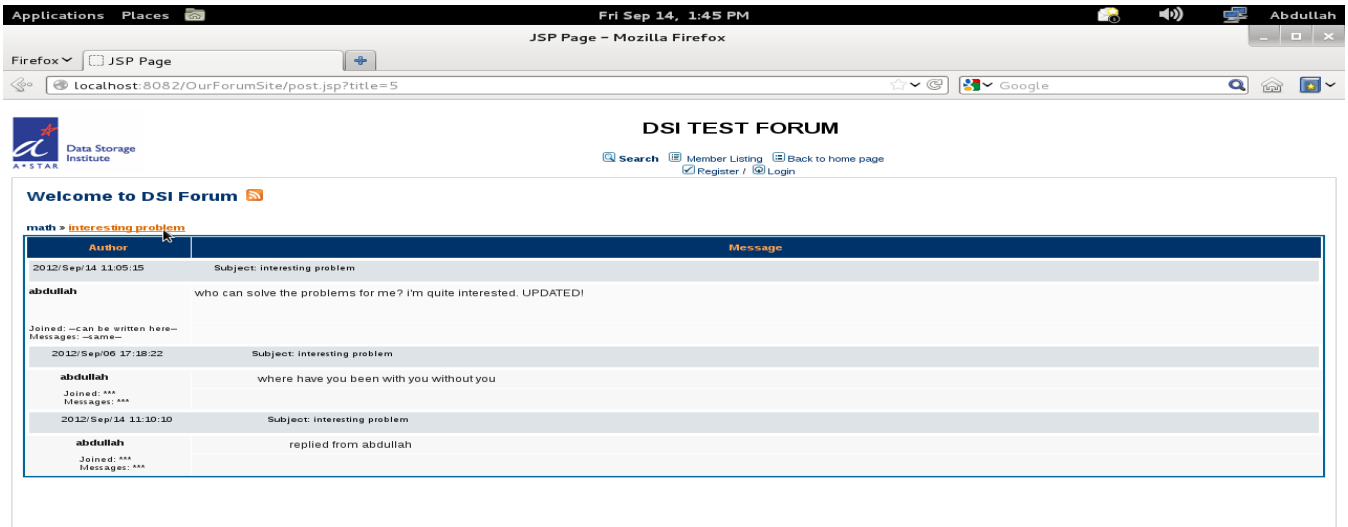


Figure 24. Posts' Page(if you not logged in)

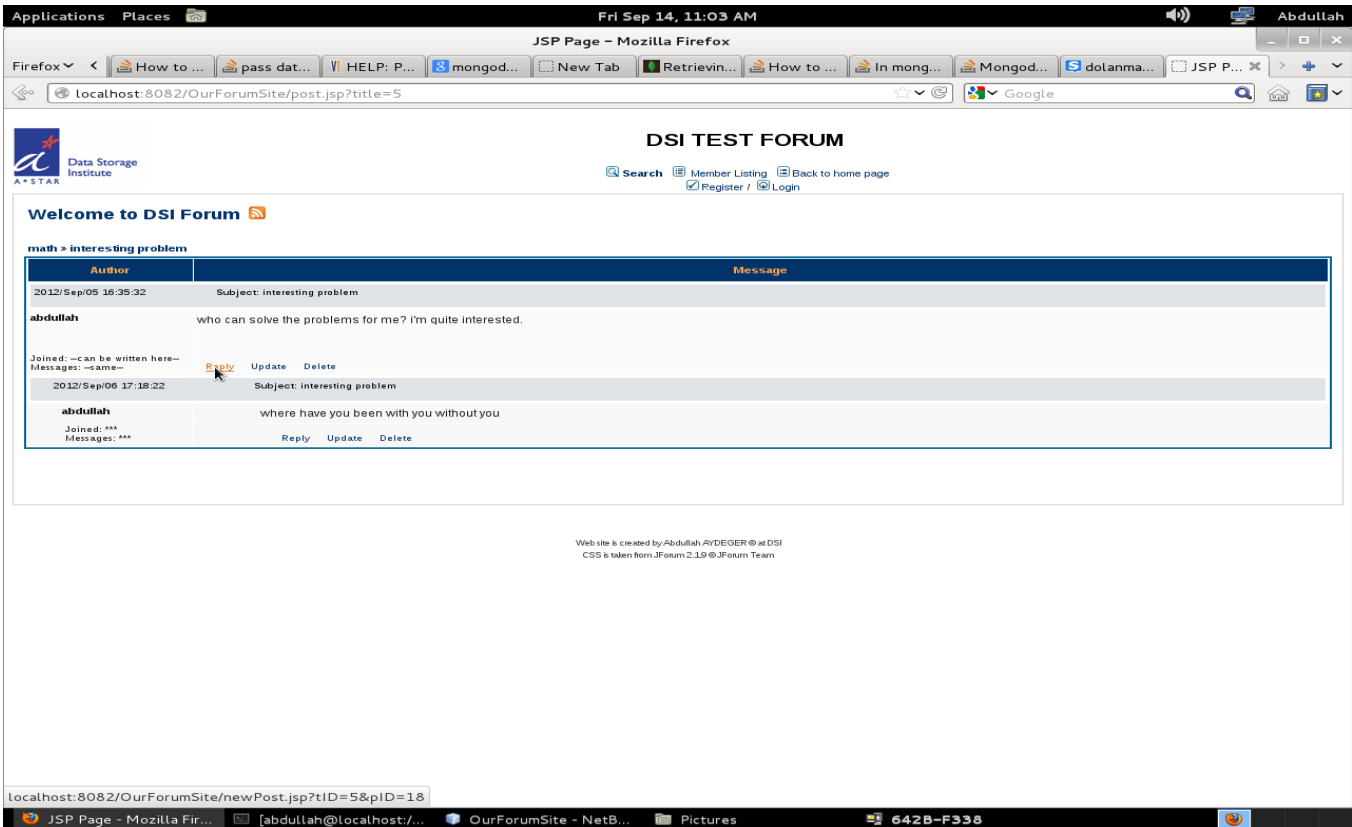


Figure 25. Posts' Page(if you logged in)

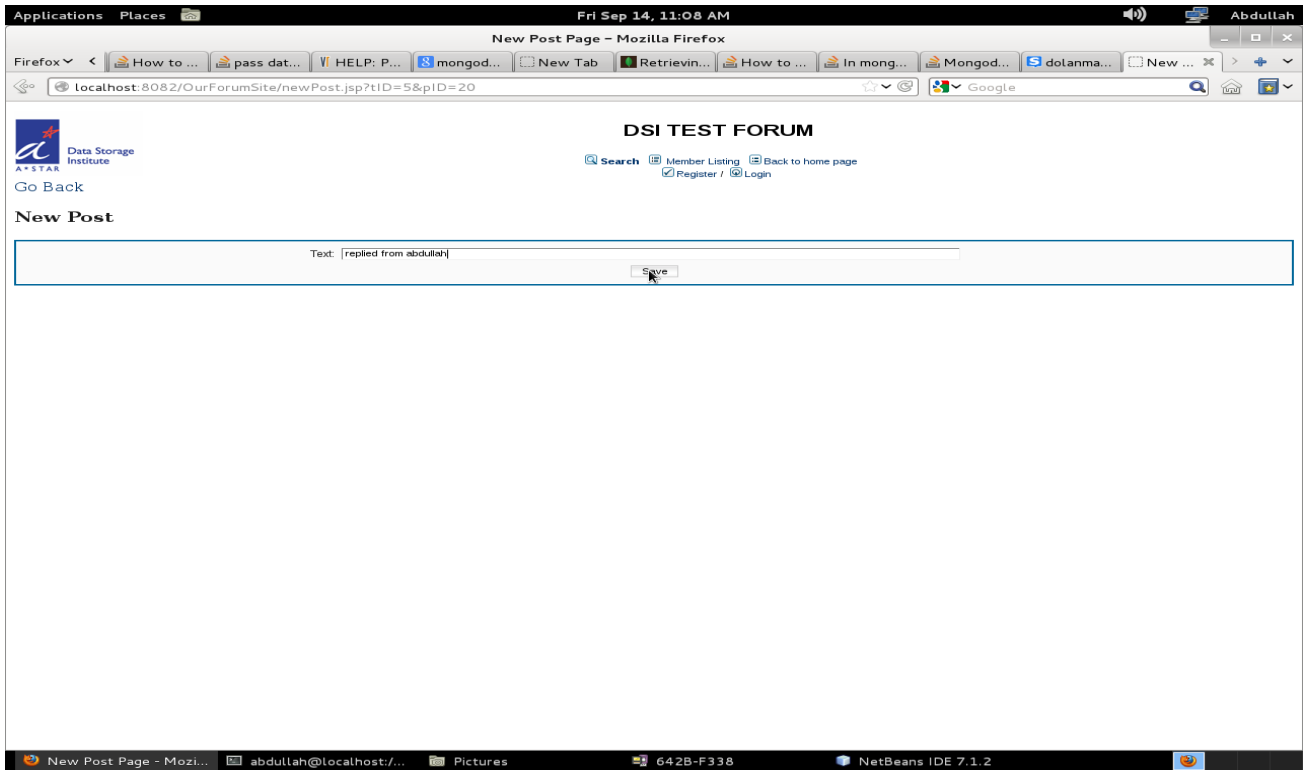


Figure 26. Page of new post addition(as a reply) for logged users

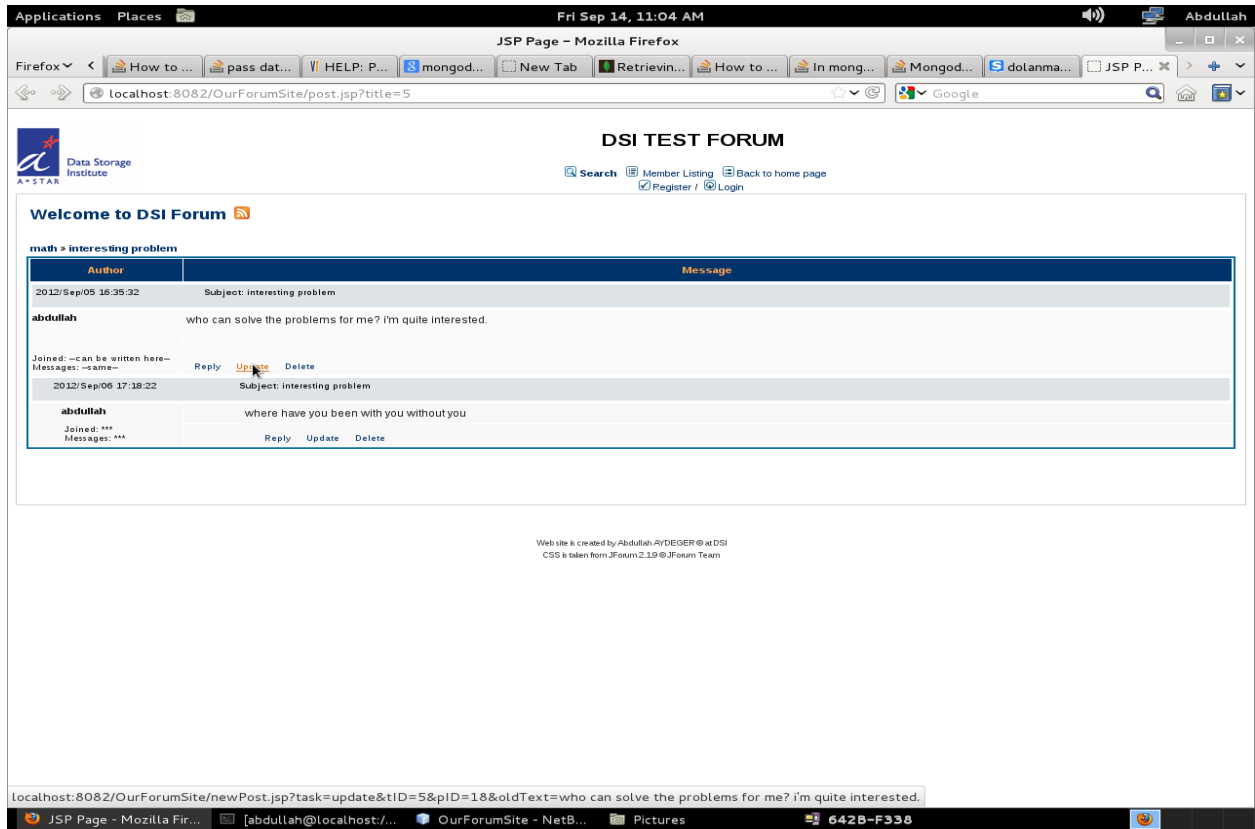


Figure 27. Clicking update link

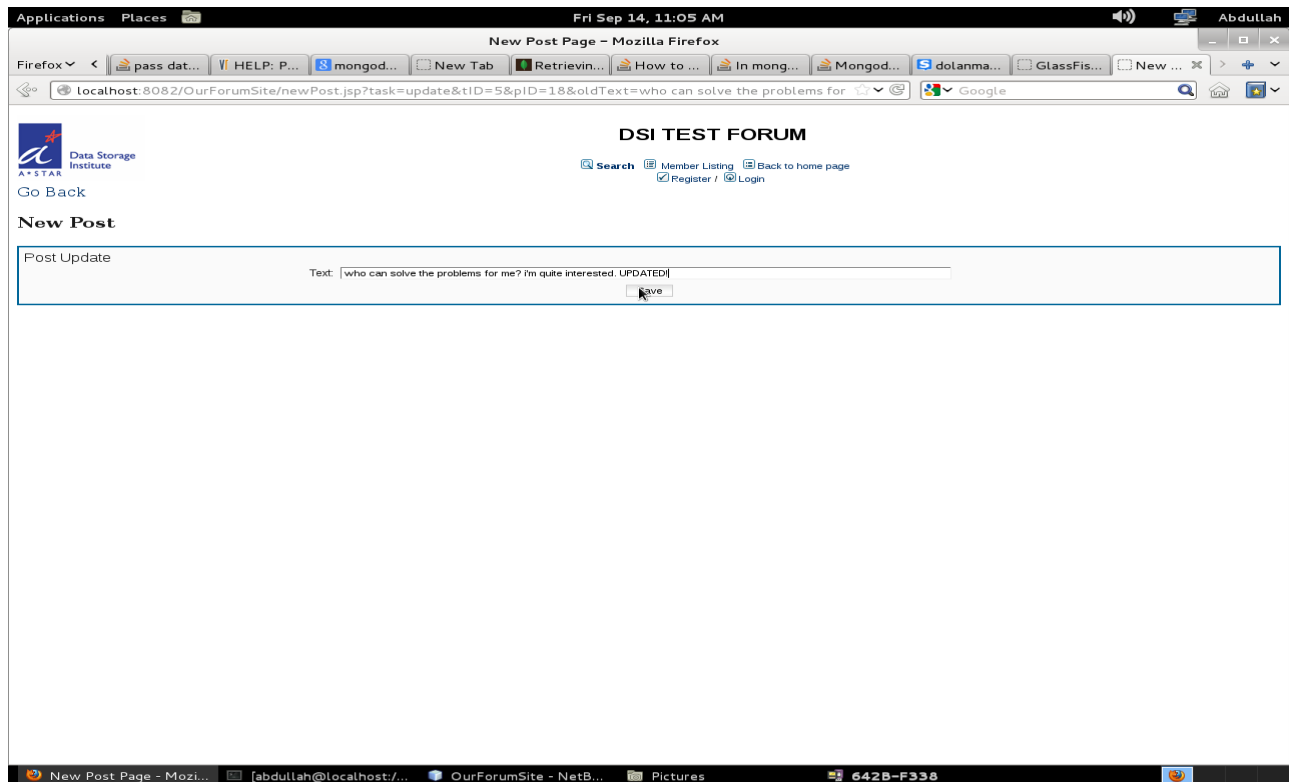


Figure 28. Update screen for post


Applications Places Fri Sep 14, 11:07 AM

JSP Page - Mozilla Firefox

Firefox Abdullah

How to ... pass dat... HELP: P... mongod... New Tab Retrievin... How to ... In mong... Mongod... dolanma... JSP P... x

localhost:8082/OurForumSite/post.jsp?title=5 Google



DSI TEST FORUM

[Search](#)
[Member Listing](#)
[Back to home page](#)
[Register / Login](#)

Welcome to DSI Forum

math » interesting problem

Author	Message
2012/Sep/14 11:05:15	Subject: interesting problem
abdullah	who can solve the problems for me? i'm quite interested. UPDATED!
Joined: --can be written here-- Messages: --same--	Reply Update Delete
2012/Sep/06 17:18:22	Subject: interesting problem
abdullah	where have you been with you without you
Joined: *** Messages: ***	Reply Update Delete

Website is created by Abdullah AYDEGER @ DSI
 CSS is taken from JForum 2.1.9 @ JForum Team

JSP Page - Mozilla Fir...
[abdullah@localhost:/...]
Pictures
6428-F338
NetBeans IDE 7.1.2

Figure 29. Updated version

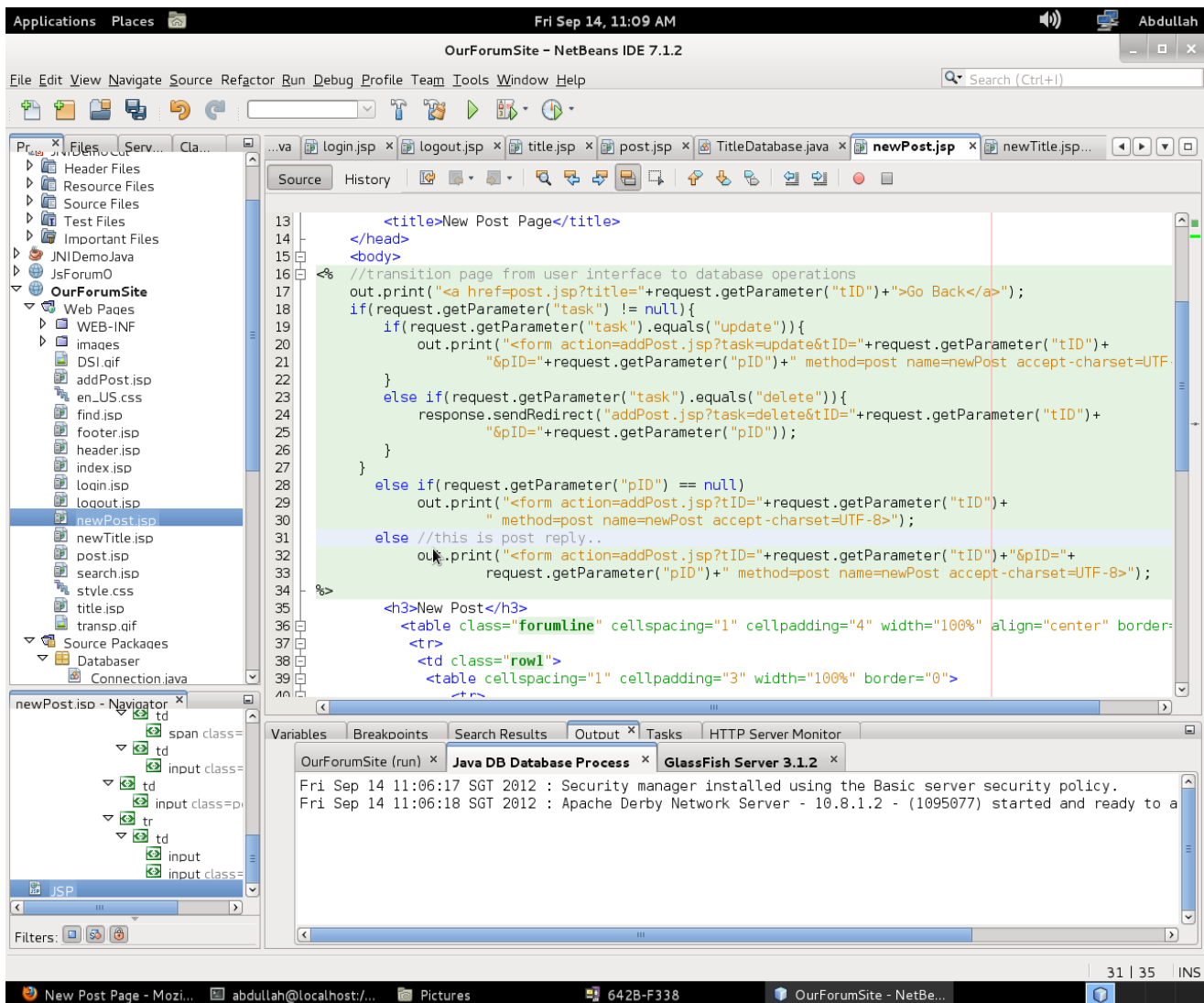


Figure 30. Determining Reply, Update and Delete operations direction(which page will be called)

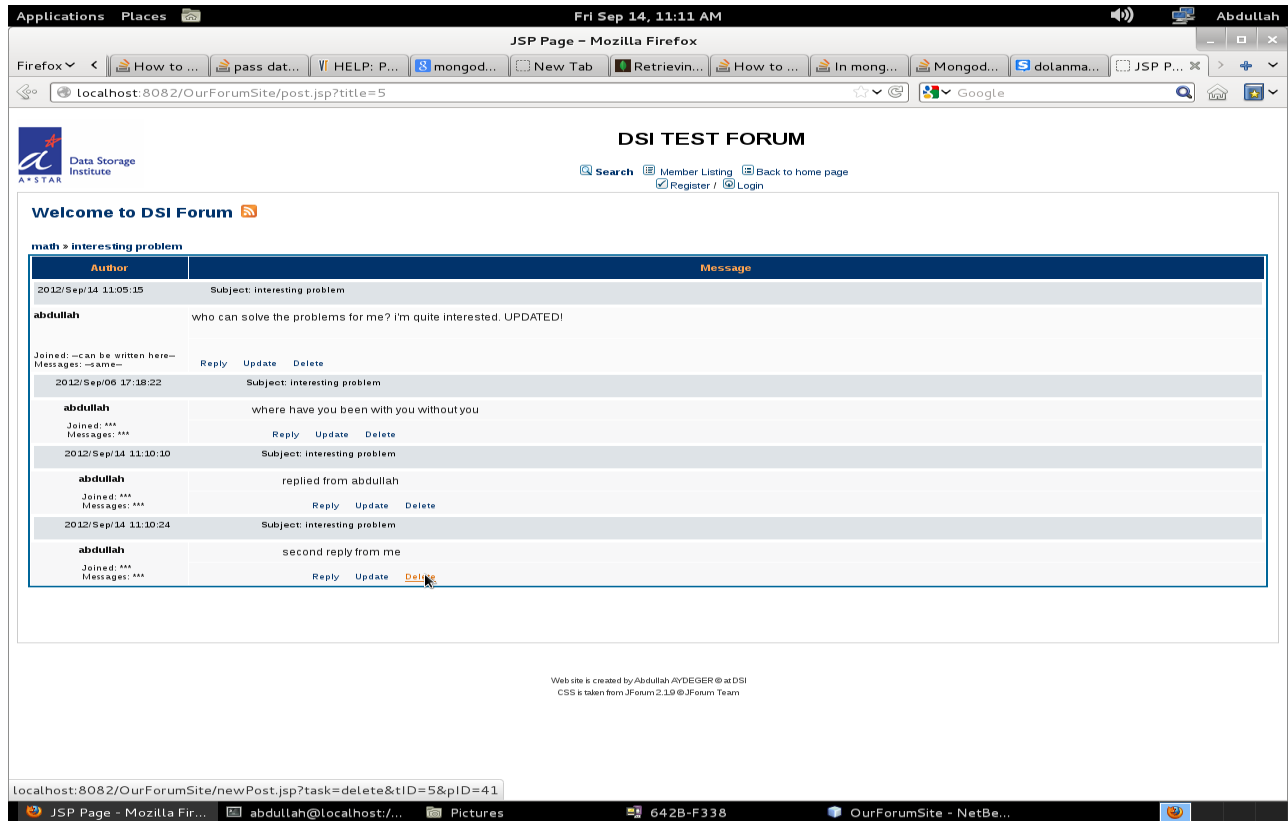


Figure 31. Clicking delete link

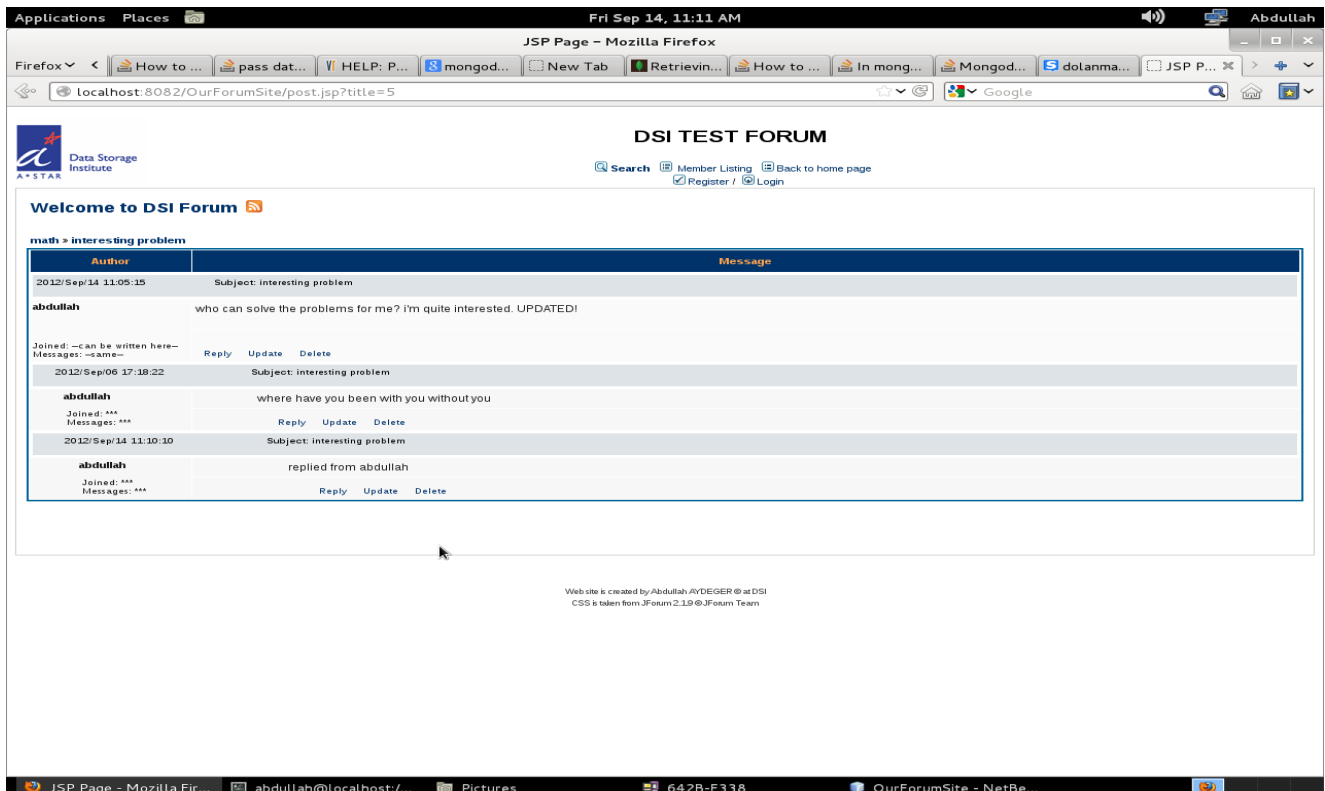


Figure 32. After clicking delete link

In the additions, deletions and updates page such as post addition, title addition some different operations will be done respectively according to clicked link. Even though they all have same page, they do different operations (all of them are implemented in addPost.jsp). These operations are determined as follow cases.

```

if(request.getParameter("task") != null){ //update or delete
    if(request.getParameter("task").equals("update")){
        BasicDBObject obj = PostDatabase.findPostByID(Integer.parseInt(request.getParameter("pID")));
        Post p = new Post(request.getParameter("postText"), session.getAttribute("user").toString(),
            request.getParameter("tID"), obj.getInt("postLevel"), obj.getInt("postID"));
        PostDatabase.updatePost(p); //same post id, other features are different

        JNIDemoJava j = new JNIDemoJava();
        j.deleteIndex(Integer.parseInt(request.getParameter("pID")), obj.getString("text"));
        //first delete old index terms from indexTable
        j.main(p.getText(), p.getPostID(), Integer.parseInt(p.getTitleID()), p.getPostLevel());
        //second add new index terms to indexTable
        response.sendRedirect("post.jsp?title="+p.getTitleID());
    }
    else if(request.getParameter("task").equals("delete")){

```

Figure 33. Update operations

```

else if(request.getParameter("task").equals("delete")){
    Object[] replies = PostDatabase.getReplies(request.getParameter("pID"));
    PostDatabase.deleteReplies();

    JNIDemoJava j = new JNIDemoJava();
    BasicDBObject obj2 = PostDatabase.findPostByID(Integer.parseInt(request.getParameter("pID")));

    j.deleteIndex(Integer.parseInt(request.getParameter("pID")),
        obj2.getString("text")); //delete itself from indexTable

    String topic = TopicDatabase.getTopicName(obj2.getString("titleID"));
    for(int k=0; k< replies.length; k++){
        BasicDBObject obj = PostDatabase.findPostByID(Integer.parseInt(replies[k].toString()));
        PostDatabase.deletePost(Integer.parseInt(replies[k].toString()));
        PostDatabase.deletePostSubpost(Integer.parseInt(obj.getString("parentID")),
            Integer.parseInt(replies[k].toString())); //deletion from postSubpost table

        j.deleteIndex(Integer.parseInt(replies[k].toString()), obj.getString("text")); //deletion from indexTable
        PostDatabase.deleteAllPostSubpost(Integer.parseInt(replies[k].toString()));
    }
    if(Integer.parseInt(obj2.getString("postLevel")) == 0){ //if it is first post of title

        if(Integer.parseInt(obj2.getString("postLevel")) == 0){ //if it is first post of title
            TitleDatabase.deleteAllTitlePost(obj2.getString("titleID"));
            PostDatabase.deletePost(Integer.parseInt(request.getParameter("pID")));
            response.sendRedirect("title.jsp?topic="+topic);
        }
        else{
            PostDatabase.deletePostSubpost(Integer.parseInt(obj2.getString("parentID")),
                Integer.parseInt(request.getParameter("pID"))); //delete from postSubpost table in parents row
            PostDatabase.deletePost(Integer.parseInt(request.getParameter("pID")));
            if(Integer.parseInt(obj2.getString("postLevel")) == 2)
                TitleDatabase.deleteTitlePost(obj2.getString("titleID"), obj2.getString("postID"));

            PostDatabase.deleteAllPostSubpost(obj2.getInt("postID"));
            response.sendRedirect("post.jsp?title="+obj2.getString("titleID"));
        }
    } // end of if phrase of deletion

```

Figure 34. Deletion operations

```

else if(request.getParameter("topic") != null){ //adding title
    if(request.getParameter("postTitle")!=null && request.getParameter("postText")!= null
        &&request.getParameter("postTitle")!= "" && request.getParameter("postText")!= ""){
        //true entrance
        Title t = new Title(request.getParameter("postTitle"), request.getParameter("topic"));
        Integer titID = (int) TitleDatabase.insertTitle(t);

        String tText = t.getTitle();
        int tLen = tText.length();
        Object[] ta = Stemmer.main(request.getParameter("postTitle"));
        Post p = new Post(request.getParameter("postText"), session.getAttribute("user").toString(),
            titID.toString(), 0);
        int pID = PostDatabase.insertPost(p);
        TitleDatabase.insertTitlePost(p.getTitleID(), pID);
        JNIDemoJava j = new JNIDemoJava();
        j.main(p.getText(), pID, Integer.parseInt(p.getTitleID()), 1);

        response.sendRedirect("post.jsp?title="+p.getTitleID());
    }
}

```

Figure 35. Title addition operations

```

else if(request.getParameter("pID") != null){ //adding reply to any post
    //addposttable
    int lev = PostDatabase.findPostLevel(request.getParameter("pID"));
    if(lev == 0) //first post of title has '0' level
        lev = 1;
    Post p = new Post(request.getParameter("postText"), session.getAttribute("user").toString(),
        request.getParameter("tID"), request.getParameter("pID"), lev+1);
    int pID = PostDatabase.insertPost(p);
    PostDatabase.insertReply(p.getParentID(), pID);
    //PostDatabase.insertPost(new Post());
    //subpost table addition
    JNIDemoJava j = new JNIDemoJava();
    j.main(request.getParameter("postText").toString(), pID, Integer.parseInt(p.getTitleID()), p.getP
        response.sendRedirect("post.jsp?title="+request.getParameter("tID"));
}

```

Figure 36. Post addition operations

Before this selection, input text for addition or update page has this source code to supply some information for next (as above) page.

```

if(request.getParameter("task") != null){
    if(request.getParameter("task").equals("update")){
        out.print("<form action=addPost.jsp?task=update&tID="+request.getParameter("tID")+
            "&pID="+request.getParameter("pID)+" method=post name=newPost accept-charset=UTF
    }
    else if(request.getParameter("task").equals("delete")){
        response.sendRedirect("addPost.jsp?task=delete&tID="+request.getParameter("tID")+
            "&pID="+request.getParameter("pID"));
    }
}
else if(request.getParameter("pID") == null)
    out.print("<form action=addPost.jsp?tID="+request.getParameter("tID")+
        " method=post name=newPost accept-charset=UTF-8>");
else
    out.print("<form action=addPost.jsp?tID="+request.getParameter("tID")+ "&pID="+
        request.getParameter("pID)+" method=post name=newPost accept-charset=UTF-8>");

```

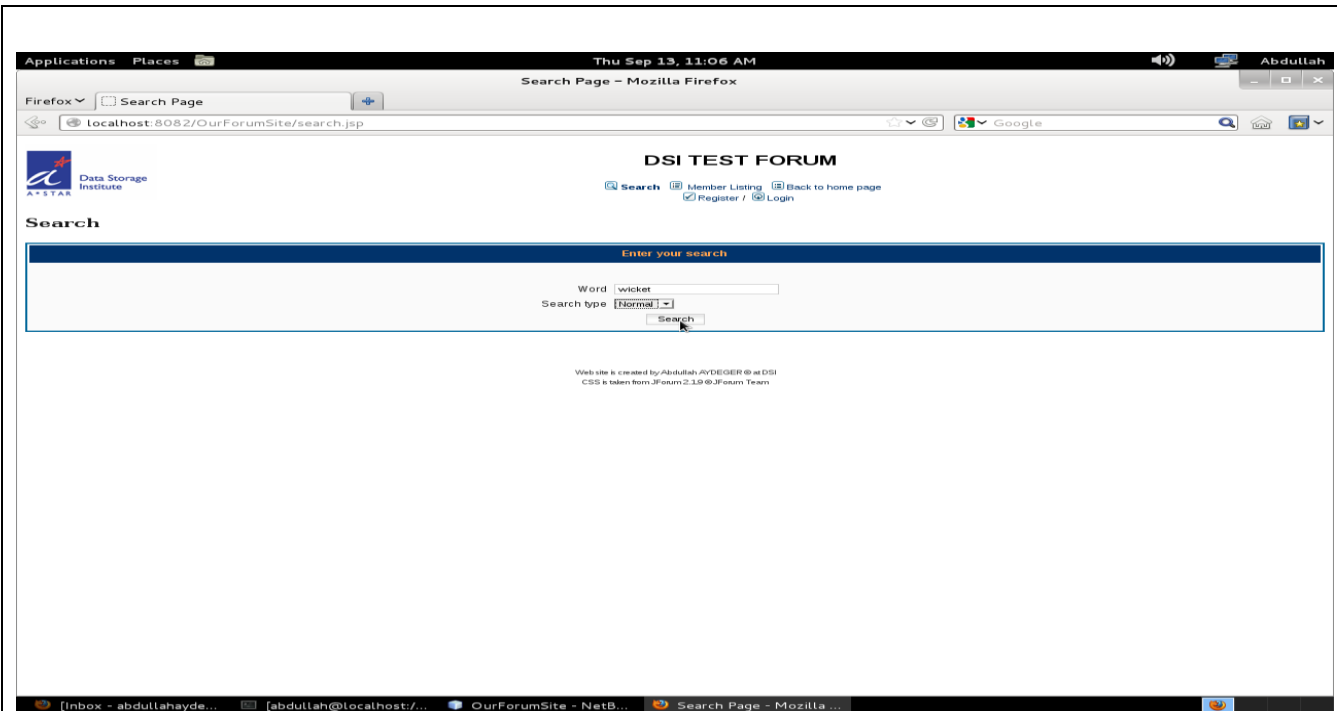


Figure 37. Search page

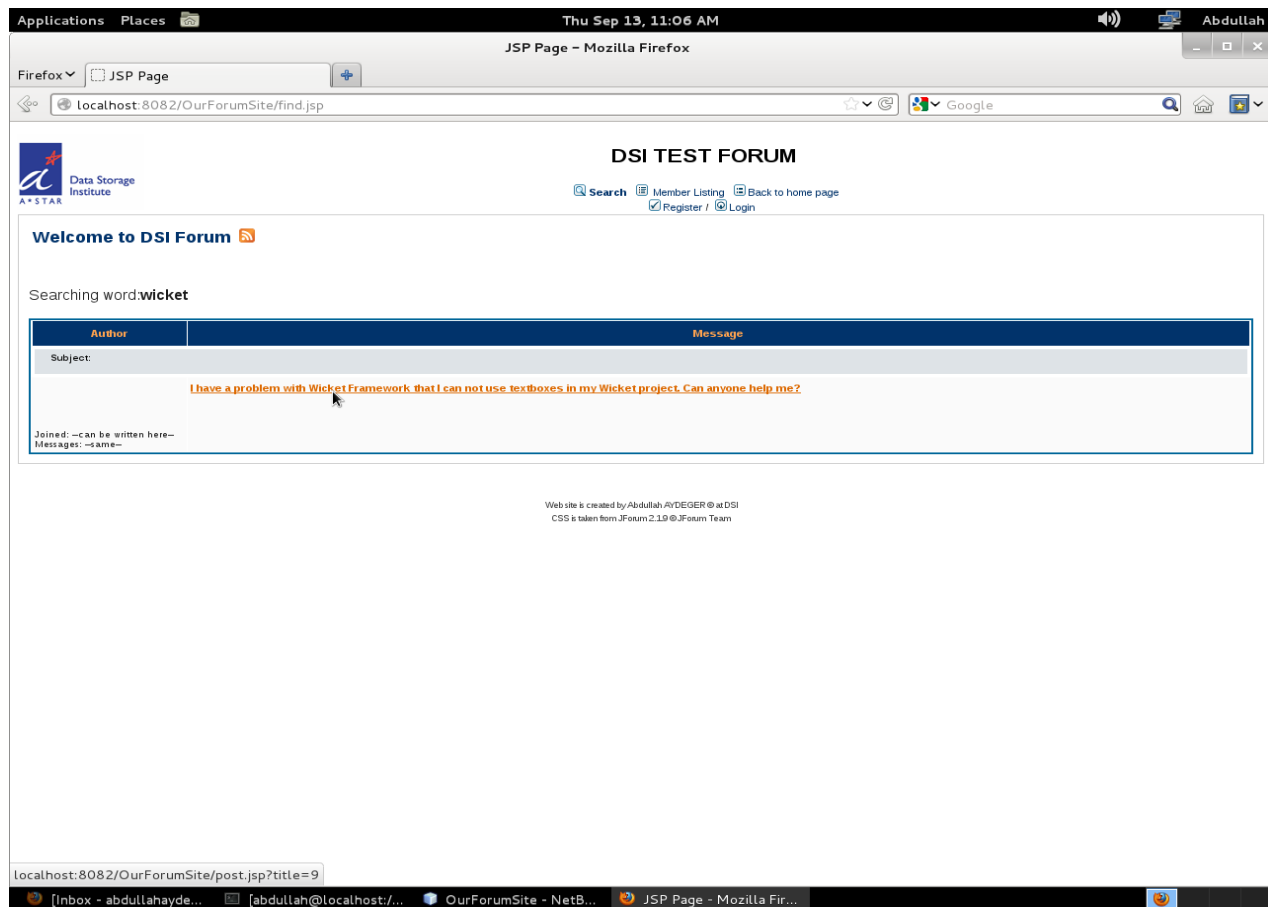


Figure 38. Search results page(find.jsp)

Which features achieved in the forum?

- Normal forum features such as post addition, update, deletion, reply
- Searching for one word by using ranking formula
- Searching more than one word
- Searching for phrases (just 2 word phrase, not more)

What else can be added?

- Ranking formula can be determined and added for more than one word
- Phrase searching can be improved for more than 2 word
- Other ranking formulas can be tested for one word searching

