



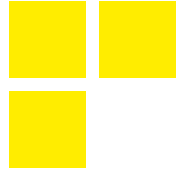
# CORETECHRA

A Technical Magazine

**MAY**  
**2019**

Vol.4 Issue 1,

# ABOUT OUR DEPARTMENT



## OUR MISSION

To produce technically competent and skilled engineers to meet the challenges of the IT industry

## OUR VISION

- Establishing innovative teaching learning practices with competent faculty and state of the art facilities.
- Enriching knowledge on latest technological advancements through industrial collaborations.
- Developing moral and ethical values through extension activities.

## PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

PEO 1:

The graduates will have a successful career in the field Information Technology and related domains.

PEO 2:

The graduates will provide solutions by applying analytical skills for the real-world problems in IT Industry.

PEO 3:

The graduates will involve in lifelong learning and as part of team in multidisciplinary projects with ethical values.

## PROGRAM SPECIFIC OUTCOMES (PSOS)

PSO 1:

Identify, formulate and solve engineering problems by applying programming concepts and algorithmic principles in the field of IT.

PSO 2:

Analyze, design and develop Software applications, Networking and Data Management technologies for efficient IT based systems.

# Contents

1. HOOT - FULL TEXT SEARCH ENGINE	01
2. CHATBOT TUTORIAL	07
3 I DON'T KNOW	13
4. List vs IEnumerable vs IQueryable vs ICollection vs IDictionary	18
5. Think Logically...,	26
6. .NET Core nginx and Postgres with EF on an rPi	27
7. Using SQLite in C#/VB.Net	31
8. Think Logically solutions	38

# HOOT – FULL TEXT SEARCH ENGINE

The smallest full text search engine (a successor for Lucent) was created from the ground up utilizing an inverted MGRB bitmap index, extremely tiny storage, and database and document modes.

## **Introduction:**

HOOT is a brand-new, incredibly compact, and lightning-fast embedded full text search engine for .net that makes use of an inverted WAH bitmap index. The majority of people are already aware with the Apache project Lucene.net, a port of the original Java version. The .net version of lucene is not maintained, and there are numerous unsupported ports of the original, which has been the subject of many complaints in the past. To get around this, I developed this project, which accomplishes the same task while being smaller, simpler, and faster. My forthcoming Raptor DB document storage database includes hOOt, which was successful that made me to decide to release as a separate entity.

## **The following articles are used by hOOt:**

Find the WAH compressed BitArray here (WAHBitArray.aspx)

- A replacement for mini Log4net is available at (<http://www.codeproject.com/KB/miscctrl/minilog4net.aspx>).
- RaptorDB's MurMur2 hash index and storage file can be obtained here (RaptorDB.aspx)
- Visit <http://www.codeproject.com/KB/IP/fastJSON.aspx> to access the fast JSON serializer.
- IFilter sans COM by Eyal Post for the example application may be found here (<http://www.codeproject.com/KB/cs/IFilter.aspx>).

Show your support for my project by using hOOt, which I will improve and enhance to full feature compatibility with lucene.net based on user feedback.

## **Why is it called "hOOt"?**

The name was inspired by the well-known Google search footer and owl logo searches (google hOOt).

## What exactly is full text search?

The act of looking for words within a block of text is known as full text searching. Full text indexers and searchers have two features:

- 1. Existence:** refers to locating words that are present among the numerous text blocks stored (for example, "bob" is present among the stored PDF documents).
- 2. Relevance,** which denotes that the text blocks retrieved are presented using a ranking algorithm that presents the most pertinent text blocks first.

The first portion is simple, the second is challenging, and there is great debate over the best ranking system to utilise. Since most programmes use and need existence more than relevance, particularly database applications, hOOt simply implements the first portion in this version.

## Why Use a Second Full Text Indexer?

- Anybody who has worked with lucene.net will confirm that it is a complicated and convoluted piece of code. I was always fascinated by how Google searches in general and lucene indexing approach and its internal algorithms, but it was just too difficult to follow. While some people attempt to produce a version that is more.net optimised, the reality is that with that code base, it is difficult to do so. The fact that it hasn't been completely redone astounds me. hOOt is a lot more straightforward, compact, and quick than lucene.net.
- Full text search on string columns in Raptor DB, the document store version, was one of the purposes behind the creation of hOOt. Since hOOt is much simpler to comprehend and modify than lucene.net, maybe more people will be able to utilise and enhance it.



## Features:

The following attributes were taken into consideration when designing hOOt:

- Extremely rapid operation speed (see performance test section)
- Extremely compact code size.
- Information is stored using BitArrays that use WAH compression.
- Multi-threaded implementation allows for simultaneous indexing and querying.
- A 38kb DLL is incredibly little (lucene.net is about 300kb).
- Storage that is well optimised and often 60% smaller than lucene.net (the more in the index the greater the difference).
- The AND operator parses query strings for spaces (e.g. all words must exist) In queries, wildcard characters (\*,?) are supported.
- By default, OR operations are performed (like lucene).
- A (+) prefix is required for AND operations (like lucene).
- A (-) prefix is required for NOT operations (like lucene).

## Limitations:

This release contains the following restrictions:

- The maximum length of a file path in document mode is 255 bytes, or its UTF8 equivalent.
- Currently, exact strings are not supported (e.g. "alice in wonderland").
- Queries do not presently support parenthesis.
- At this time, relevance and ranking are not supported.
- User specified document field searches are not supported at this time.



## Performance Tests:

On my laptop PC, tests were run under the following conditions: Windows 7 Home Premium 64-bit, AMD K625 1.5GHz, 4GB DDRII RAM, and a Win Index of 3.9. hOOt generates a large amount of log data that you may use to monitor engine activity. The following are the results of a crawl I conducted using the sample application over a set of approximately 4600 files totaling about 5.8Gb of data in various formats (keep in mind that the application was built using .net 4 and is running on a 64-bit operating system and that 64-bit IFilters were also installed):

Document Count	4,490
Total time	~22 min
Index file size	29 Mb
Text size total	632,827,458 characters
Total hOOt Indexing time	56767.2471 ms ~ 56.7 secs
Total hOOt writing document info time (fastJSON serialize)	110632.3282 ms ~ 110 secs
Total words in hOOt	~290,000

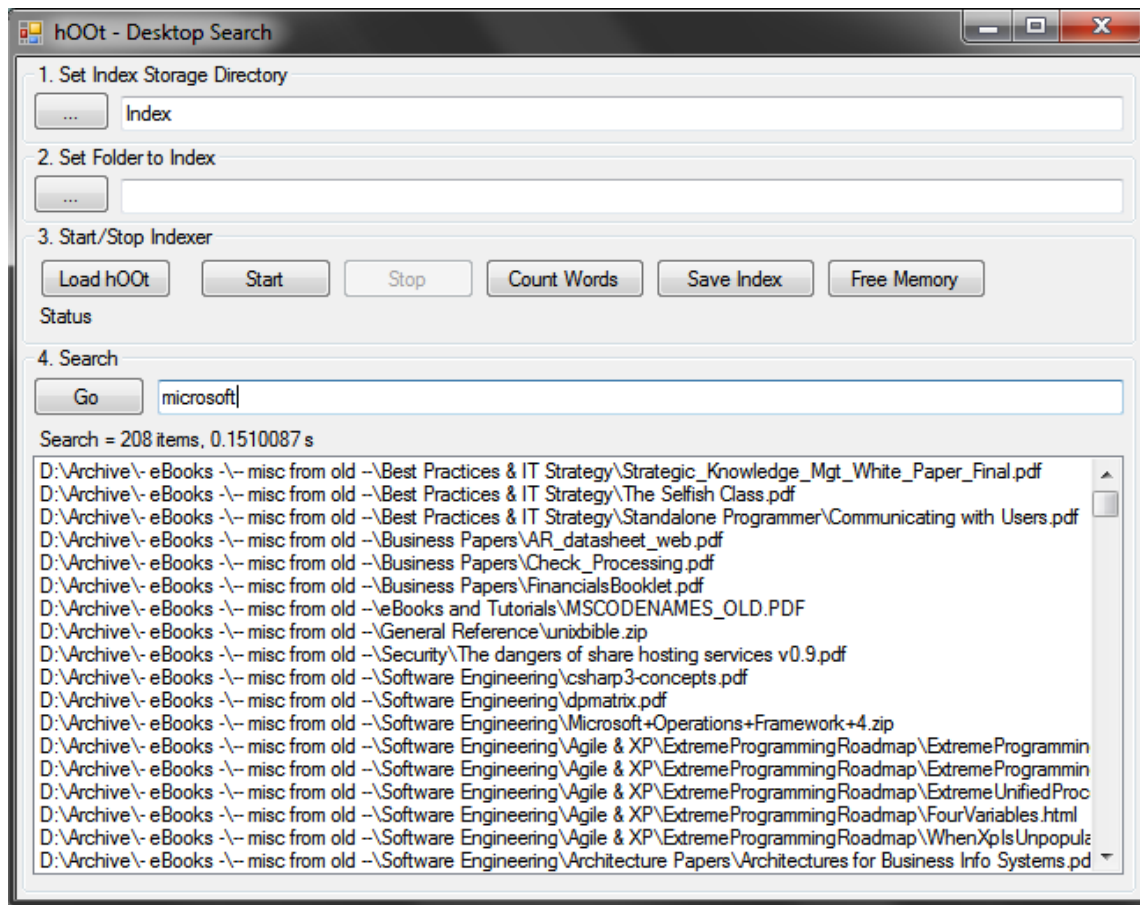
As you can see, the indexing engine is quite quick, processing 632 MB of text in just 57 seconds. The IFilter extraction time accounts for the remaining 22 minutes of the overall time. On the engine side, all queries for the aforementioned document count complete in roughly 1 milliseconds, however as you can see in the sample image, a search for "microsoft" returned 208 results in 0.151 seconds, which is again a result of the document deserialization time. By comparison lucene.net has the following:

Index file size	70.5 Mb
Total time	~28 min

## Performance Tests v1.2:

The size of the index is decreased to 24Mb, or around 19% less space, by using a better word extractor.

## Using the Sample Application:



The standard desktop search programme created using hOoT is shown above. Follow these steps to utilise the application:

1. In the (1) group box, set the index storage directory, which will store all index data.
2. In the (2) group box, select the folder where you want to crawl content.
3. You can complete the following in the (3) group box:
  1. Load hOoT: This loads hOoT, allowing you to query an already-existing index.
  2. Start Indexing: After clicking this, hOoT will load and begin indexing the directory you specified.
  3. Stop Indexing: This option becomes available once indexing has begun, allowing you to halt the procedure.
  4. Word Count: This will reveal how many words are in the HOOts dictionary.
  5. Save Index: By using this, whatever in memory will be saved to disc.



6. Internal free memory method on hOOt will be invoked by this (this will only free the bitmap storage and not release the cache).
7. You can use the (4) group box to search for material; the label will display the number of results and the search time.
8. Simply double-click the file path in the list box to open the file.

This is only a demonstration of hOOt; while it functions as is, a real application would benefit from more features. Please install the IFilter handlers listed below before using this sample for best results:

- This website (<http://www.foxitsoftware.com>) offers the Foxit PDF filter. [Adobe is terrible to use since it is so slow]
- You can download the Microsoft Office 2010 IFilter from this link (<http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=17062>).



**ARUN BHARATHI M**  
**II-IT**

# CHATBOT TUTORIAL

## Introduction:

A chatterbot is essentially a computer software that, when given Natural Language inputs (such as English or French), reacts by saying something significant in that same language. Thus, the effectiveness of a chatterbot might be evaluated by the standard of the output it chooses to provide in response to the user. We could infer from the previous explanation that a very rudimentary chatterbot could be created in a few lines of code using a certain programming language. Let's build our first chatbot. Note that all of the C++ code used in this tutorial will be used. The reader's familiarity with the STL library is also assumed.) The following languages are also supported by this tutorial [Java](#), [Visual Basic](#), [C#](#), [Pascal](#), [Prolog](#) and [Lisp](#).

```
// Program Name: chatterbot1
// Description: This is a very basic example of a
// chatterbot program//
// Author: Gonzales Cenelia//
#include <iostream>
#include <string>
#include <ctime>
int main()
{
    std::string Response[] = {"I HEARD YOU!", "SO,
    YOU ARE TALKING TO ME.", "CONTINUE,
    I'M LISTENING.", "VERY INTERESTING
    CONVERSATION.", "TELL ME MORE..." };
    srand((unsigned) time(NULL));
    std::string sInput = "";
    std::string sResponse = "";
    while(1) {
        std::cout << ">";
        std::getline(std::cin, sInput);
        int nSelection = rand() % 5;
        sResponse = Response[nSelection];
        std::cout << sResponse << std::endl;
    }
    return 0;
}
```

As you can see, writing a very basic programme that can interact with a user doesn't require a lot of code, but it would probably be quite challenging to develop a programme that would truly be able to understand what the user is actually saying and would then generate an appropriate answer to it. Since the outset, even before the first computers were made, they have been a long-term objective. The British mathematician Alan Turing presented the question, "Can machines think?," and he also suggested the Turing Test, which is today used to determine if a machine can think.

In this test, the judge must determine which of two speakers—a computer programme and a human person—is speaking to him or her in real time. These days, there is a contest called the Loebner Prize, where bots that successfully deceived the majority of the judges for at least five minutes would win a prize of \$100,000. No computer programme has so far been able to pass this test. One of the main causes of this is that computer programmes created for such competitions have a natural inclination to make a lot of typographical errors (they are often out of the context of the conversation).



Therefore, determining whether a judge is speaking to a "computer programme" or a human person is typically not that tough. Eliza, a programme created in 1966 by MIT professor Joseph Weizenbaum, is also the immediate ancestor of all systems that attempt to simulate a dialogue between actual people.



In contrast to strong artificial intelligence, which aims to develop algorithms that are at least as intelligent as humans, chatbots are often seen as belonging to the weak AI sector. However, this does not imply that chatbots are without real potential. A significant development for the study of AI would be the ability to develop software that could communicate in a manner similar to that of humans. Chatbots are a kind of artificial intelligence that are easier for hobbyists to use (it only takes some average programming skill to be a chatbot programmer). Thus, creating intelligent chatbots is an excellent location for programmers who want to construct genuine AI or some other form of artificial intelligence to start.

There are a lot of them, I suppose. First of all, it is very evident that the computer is simply choosing a random response from his database each time the user types a sentence on the keyboard, rather than actually attempting to understand what the user is saying. Additionally, we can see that the programme repeats itself quite a bit. One of the causes of this is because the database is quite limited in size (5 sentences). The absence of any mechanism to regulate this undesirable conduct is the second factor that could account for the repetitions.

**How can we change a programme from one that randomly chooses responses to whatever input the user might type on the keyboard to one that demonstrates a greater comprehension of the inputs?**

We just need to use keywords, which is a very straightforward solution to the problem.

A keyword is simply a sentence (not necessarily a complete one) or even a single word that the programme may identify from the user's input and respond to (example: by printing a sentence on the screen). The knowledge base or database for the upcoming application will be made of keywords and some responses related to each keyword. We now know what needs to be done to make "our first chatterbot" better and smarter. Let's write "our second bot," which we'll call chatterbot2.

Program Name: chatterbot2	Contd.,	Contd.,
<pre>// Description: this is an improved version  // of the previous chatterbot program "chatterbot1"  // this one will try a little bit more to understand what the user is trying to say  // Author: Gonzales Cenelia  #pragma warning(disable: 4786)  #include &lt;iostream&gt;  #include &lt;string&gt;  #include &lt;vector&gt;  #include &lt;ctime&gt;  const int MAX_RESP = 3;  typedef std::vector&lt;std::string&gt; vstring;  vstring find_match(std::string input);  void copy(char *array[], vstring &amp;v);  char *responses[MAX_RESP]; }record;</pre>	<pre>typedef struct {      char *input;      break;  }  else if(responses.size() == 0) {      std::cout &lt;&lt; "I'M NOT SURE IF I UNDERSTAND WHAT YOU ARE TALKING ABOUT."      &lt;&lt; std::endl;  }  else {      int nSelection = rand() % MAX_RESP;      sResponse = responses[nSelection]; std::cout &lt;&lt; sResponse &lt;&lt; std::endl;  } }  return 0;}  // make a search for the user's input // inside the database of the program</pre>	<pre>vstring find_match(std::string input) {  vstring result;  for(int i = 0; i &lt; nKnowledgeBaseSize; ++i)  {  if(std::string(KnowledgeBas e[i].input) == input) {  copy(KnowledgeBase[i].res ponses, result);  return result;  } }  return result;  }  void copy(char *array[], vstring &amp;v) {  for(int i = 0; i &lt; MAX_RESP; ++i) {  v.push_back(array[i]);  } }</pre>

At this point, the algorithm is capable of comprehending phrases like "what is your name," "are you intelligent," etc. Additionally, he has the option to select the most relevant answer from his list of options for the given statement and just put it on the screen. Chatterbot2 is able to select an appropriate response to the user input given, as opposed to Chatterbot1, which would randomly select responses without considering what the user is actually attempting to say.

These new programmes also include a handful of new strategies. For example, when the software cannot discover a term that matches the current user input, it simply responds by saying that it doesn't understand, which is very human-like.

### **What Can We Improve on these Previous Chatbots to Make It Even Better?**

There are many areas where we might make improvements. The first is that since the chatterbot has a tendency to repeat itself a lot, we might develop a system to limit these repetitions. To check whether the next bot response is different from the prior one, we could simply store the previous response of that Chatbot in a string called sPrev Response. If this is the case, we then choose a different response from those that are offered.

The chatbot's handling of user inputs is another area where we may make improvements. At the moment, even if your input is in lower case and there is a match in the bot's database, it will be completely ignored by the chatbot. Additionally, if the input contains extra spaces or punctuation (!,., ), the Chatbot will not be able to read it.

Because of this, we'll aim to implement a fresh way to preprocess user inputs before they're looked up in the chatbot database. Since the database's keywords are all capitalised, we could create a function that converts user input to upper case, and another method that simply removes any punctuation or extra spaces that may be present in user input. Having stated that, we now have enough information to create "Chattebot3," our upcoming chatterbot.

### **What Can We Improve in Chatterbot4 to Make It Better?**

#### **Here Are Some Ideas:**

- ❖ Since the chatterbot code has begun to expand, it would be a good idea to use a class to encapsulate the next chatterbot's implementation.
- ❖ Additionally, we will need to add additional entries to the database because it is still far too little to support a true interaction with users.

- ❖ We must also deal with the possibility that a user may occasionally press the enter key without really typing anything.
- ❖ The user may attempt to deceive the chatterbot by repeating a previous sentence slightly modified; in this case, we must count it as a user repetition.
- ❖ Finally, you'll probably start to notice that when there are several keyword options for a given input, we sometimes need a mechanism to rank them so we can choose the best one.

You are recommended to attempt compiling and executing the code for "chatterbot5" before moving on to the next section of this tutorial so that you can understand how it functions and to confirm the modifications that have been made. As you may have seen, a class now contains the implementation of the "current chatterbot," and the updated version of the programme also includes several new functions.

**IZJAS AHAMED I**

**II-IT**

## I DON'T KNOW

### **The world of software development is vast.**

The realm of software development is enormous—I mean enormous. It's overwhelming having to deal with front-end people, back-end people, database specialists, architects, engineers, people who can communicate with engineers, and many other people. With the speed at which technology is developing, it can be difficult to keep up with your own environment, let alone the vast universe that revolves around you.

And for that reason, I'm writing this article. I'm writing this to reassure you that it's perfectly acceptable to lack knowledge and that, frequently, saying "I don't know" may make everyone's lives—including your own—much, much simpler.

### **Why Are You Unaware?**

The field of software development is enormous and has a very wide scope. A few examples are networking, embedded systems, front- and back-end development, machine learning, web services, architecture, CSS, and mobile development. There are so many parts of the industry that you could easily focus solely on one for your whole career. Keeping up with each of them is both impractical and difficult because they are all constantly growing with new features, languages, paradigms, and patterns.





Don't try, then.

Learn to occasionally say, "I don't know." Accept it; don't see it as a flaw but as a strength and a resource that will spare you time, effort, and perhaps heartache.

You will be one step closer to finding the solution to your issue if you swallow your pride and confess that you are in the dark about something.

Say you have just graduated from college. You are a top-of-the-class graduate of a prestigious computer science institution, have close relationships with many of the top tech companies in the nation, and are capable of writing any algorithm or resolving any programming conundrum.

Everything has been easy so far, but once you start working as an engineer, you'll see that knowing all those information is completely different from being able to handle problems in the real world. When you are asked to "change the CSS and correct an issue with the z-index," all of your hours spent on optimising red-black trees for whiteboard interviews will be for naught.



You might explain this by saying that you were just asked a question about a subject for which you weren't prepared or that you were just entering the field. When you finally mastered z-indices, you would feel like you had conquered the world—at least until the next curveball

situation of that nature reared its ugly head and reset the cycle. The message is this: don't try to keep up with software's constant evolution since it happens too quickly and in too many ways.

### **Why do you believe I don't require knowledge?**

It takes time, effort, and typically some serious practise and experimenting to learn new things. Contrary to what you might see on television and in movies, creating software is rarely a solo endeavour. Professionally, it usually doesn't matter if you know 100 programming languages, can write a merge-sort while wearing blinders, or understand what the volatile keyword in C# does. They seek someone who is capable of the following:

- Fix a dilemma
- Having fun with others
- Finish the job

I can't stress enough how important these three abilities are to a developer's day-to-day activities. Two of them might appear particularly technical, but the third, "plays well with others," is perhaps the most crucial of all and won't be covered in a classroom, on a Plural sight course, or anywhere else.



Technical proficiency is important for your profession, but it doesn't matter nearly as much as being able to use Google and get along with others. Throughout your work, you'll

probably come across several outstanding developers, both men and women, who will simply astound you with their brilliance.

It's typically because they can check those three boxes, not because they can memorise all the keywords in your favourite programming language or create their own sort, as I indicated previously.

As developers, our primary goal is to complete tasks and so provide value. Sometimes we may simply "know how" to tackle a problem, but more often than not, we slam open a new tab and land on a blog post, Stack Overflow page, or some discussion buried deep within our preferred technology. If you don't have to, concentrate on learning how to be resourceful and how to approach finding an answer rather than trying to keep everything "in memory."



## **No one else has all the information:**

- ❖ It's important to remember that if you don't know something, you're in luck because it's likely that many other people also don't know it. You are not alone, and most likely most problems you encounter aren't just the one on which you are fixated.
- ❖ Even the most talented coders will experience unproductive days and endless hours lost in the sea of pointless search results. Finding engineers of the highest level venting on Twitter about missing a keyword or command or spending the entire day struggling with a CSS issue doesn't take a lot of time. Everyone experiences it, and it comes as no surprise when you work all day developing software.
- ❖ Instead of straining to commit everything to memory, put your attention on learning how to study issues and find answers. You'll gain some sanity as well as time, effort, and mental energy.
- ❖ When you are struggling with a problem or issue but are unable to immediately identify the solution, try not to get discouraged. Take comfort in the knowledge that the majority of developers, from the newest to the most experienced, would be in your position, opening their preferred browser and frantically looking for the best solution.
- ❖ Just remember that it's totally fine if you don't know the answer since, chances are, neither do most of us.

**AFRIN BANU N**

**III-IT**

## List vs IEnumerable vs IQueryable vs ICollection vs IDictionary

### IList:

Why is IList necessary? IList, an interface that implements methods, is used by List. Use IList whenever you anticipate that your code may need to be modified in the future because it lowers dependency and allows for minimal code adjustment. In order to decouple your programme and have control over any potential method changes, you need be aware of polymorphism. The other things are comparable. Whenever we want to change an operation, "IList" makes it simple for us to do so by at the very least changing all of the codes. It should be instantiated from a list because interfaces cannot be instantiated.

### Difference between Concrete Class and Interface:

- ❖ Although a concrete class can implement ONE or MORE interfaces, it only derives from ONE class. While you only need to define the signature inside the interface, you can write the whole version of your function inside a concrete class.
- ❖ While variables cannot be defined inside interfaces, they may be defined inside concrete classes.
- ❖ While you are not allowed to define a function Object() { [native code] } inside an interface, a concrete class may have one.
- ❖ While interfaces lack access modifiers, abstract classes can.

```
//Ilist can not be instantiate from Ilist , so it should be instantiate from List
System.Collections.Generic.IList<string> strIList = new List<string>();
strIList.Add("Mahsa");
strIList.Add("Hassankashi");
strIList.Add("Cosmic");
strIList.Add("Verse");
this.ListBoxListGeneric.DataSource = strIList;
this.ListBoxListGeneric.DataBind();
System.Text.StringBuilder str = new System.Text.StringBuilder();
foreach (var item in strIList)
```

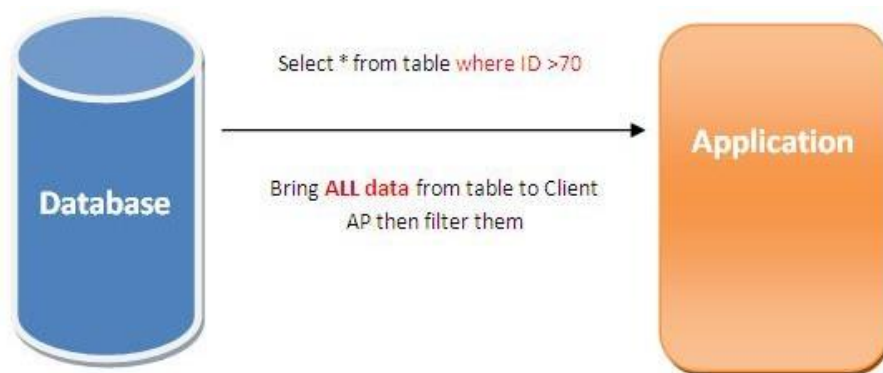
```

{
    str.Append(" , " + item);
}
this.lblList.Text = str.ToString();

```

## IEnumerable:

IEnumerable is only appropriate for iterating across collections; adding or removing data is not possible. IEnumerable assumes you have a large number of records so it places a burden on your memory before bringing ALL info from the server to the client and filtering it.



//IEnumerable can not be instantiate from Enumerable , so it should be instantiate from List

```

System.Collections.Generic.IEnumerable<Employee> empIEnumerable = new
List<Employee>

```

```

{ new Employee { ID = 1001, Name="Mahsa"},
  new Employee { ID = 1002, Name = "Hassankashi" },
  new Employee { ID = 1003, Name = "CosmicVerse" },
  new Employee { ID = 1004, Name = "Technical" }

```

```

};

```

```

this.GridViewIEnumerable.DataSource = empIEnumerable;

```

```

this.GridViewIEnumerable.DataBind();

```

```

System.Text.StringBuilder str = new System.Text.StringBuilder();

```

```

foreach (Employee item in empIEnumerable)

```

```

{

```

```
str.Append(" , " + item.ID + "-" + item.Name);  
}  
this.lblIEnumerable.Text = str.ToString();
```

### **IQueryable:**

When dealing with large datasets including a large number of records, we must minimise application overhead. In these circumstances (large amounts of data), IQueryable prepares good performance by filtering data first and sends filtered data to the client.

```
//Difference between IQueryable and IEnumerable  
//You can instantiate IEnumerable from List  
IEnumerable<employee> queryIEnumerable = new List<employee>() ;  
//Bring ALL records from server --> to client then filter collection  
//To bring all data from server you should omit where clause from linq to sql  
queryIEnumerable = from m in ctx.Employees select m;  
//If you use where as extension method with IEnumerable then All records will be  
loaded  
queryIEnumerable = queryIEnumerable.Where(x => x.ID == 1).ToList();  
//You cannot instantiate IQueryable  
IQueryable<employee> queryIQueryable=null;  
//Bring just ONE record from server --> to client  
queryIQueryable = (from m in ctx.Employees where m.ID == 1 select m);  
//Whenever you call IQueryable so ==> It will be executed  
this.GridViewIQueryable.DataSource = queryIQueryable.ToList();  
this.GridViewIQueryable.DataBind();
```

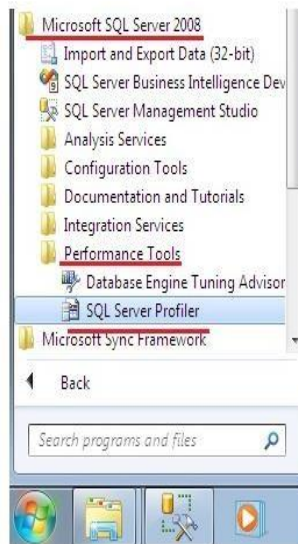
### **SQL Profiler:**

How to See How Many Records Will Be Loaded & How Your Query Generates TSQL?



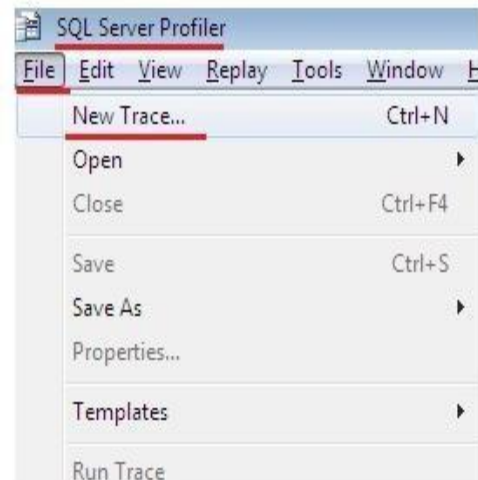
### Step 1

Start -> MS SQL Server 2008 -> Performance Tools -> SQL Server Profiler



### Step 2

SQL Server Profiler -> File -> New Trace



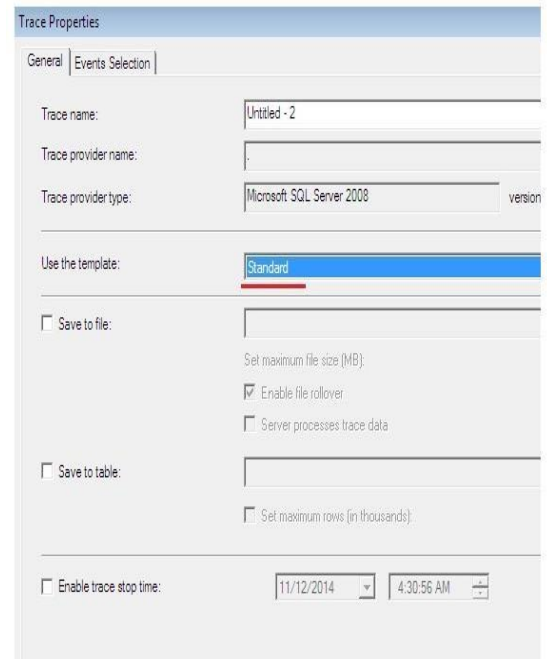
### Step 3

Connect with your user name and password.



### Step 4

General (Tab) -> Use the Template: Standard



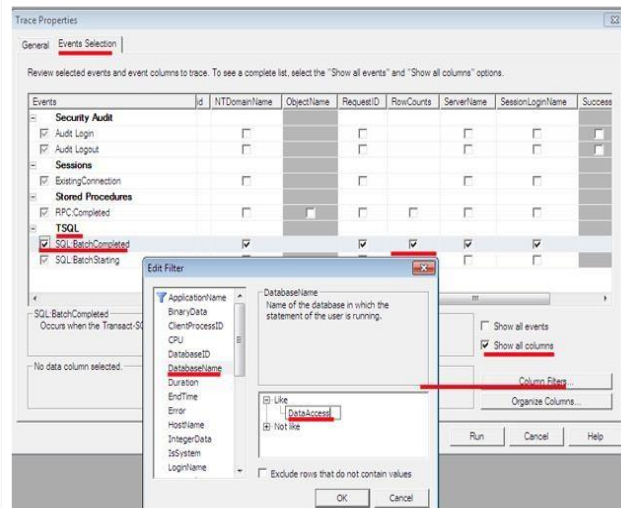


## Step 5

Event Selection (Tab) -> Event: TSQL -> Select: SQL-BatchCompleted | Select Show all Columns

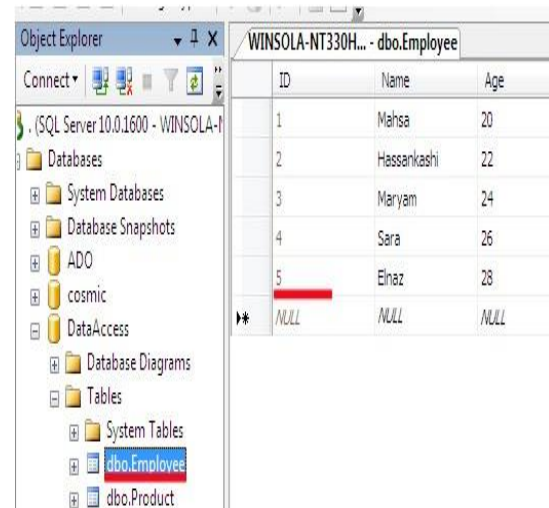
Press Column Filter -> Database Name: Like: "DataAccess"

Press Run.



## Step 6

Go To MS SQL Server Management Studio -> Count all of records (records=5)

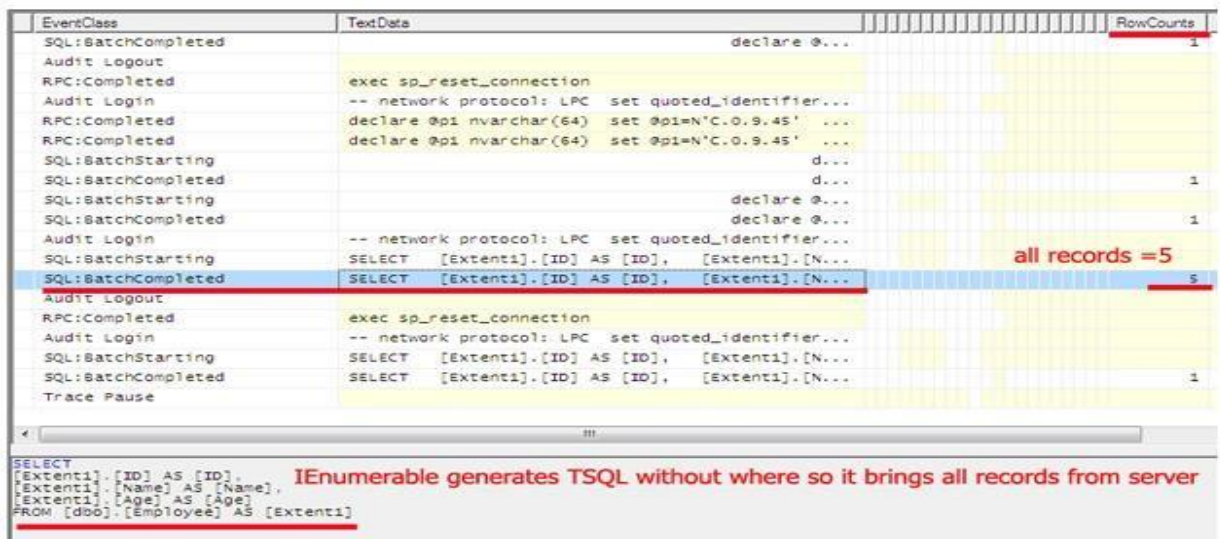


## Step 7

### IEnumerator Generates

#### SQL

```
SELECT
[Extent1].[ID] AS [ID],
[Extent1].[Name] AS [Name],
[Extent1].[Age] AS [Age]
FROM [dbo].[Employee] AS [Extent1]
```



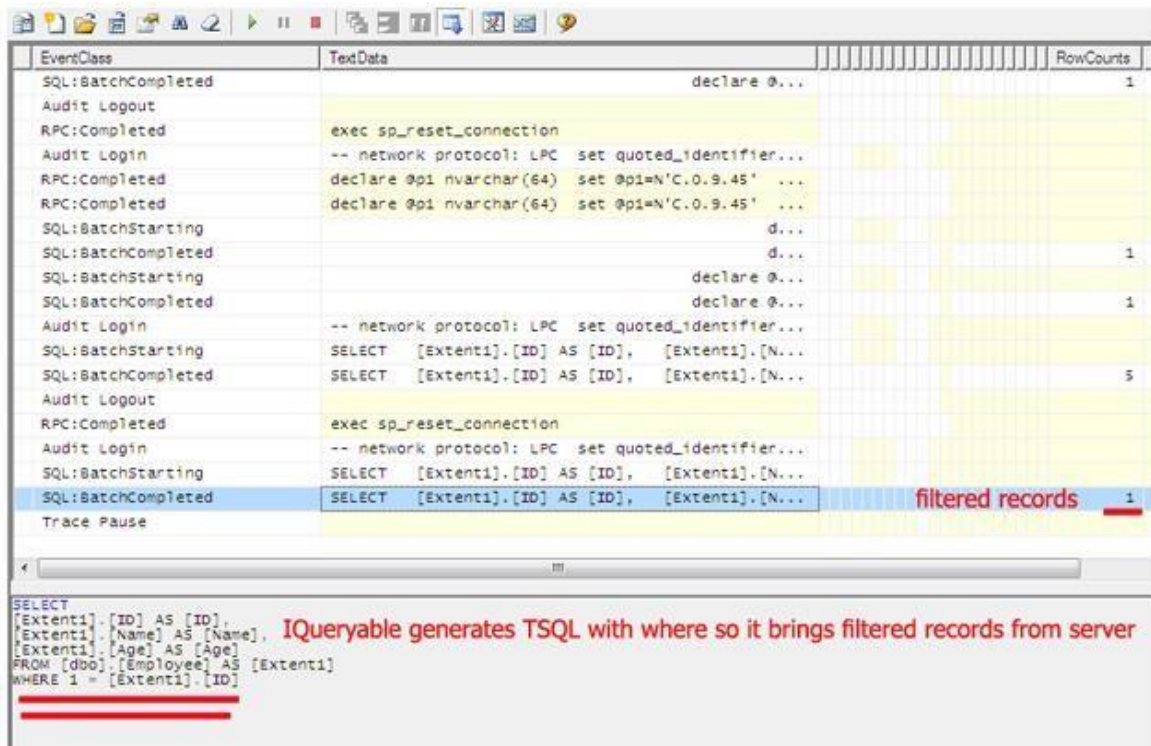
SELECT  
[Extent1].[ID] AS [ID],  
[Extent1].[Name] AS [Name],  
[Extent1].[Age] AS [Age]  
FROM [dbo].[Employee] AS [Extent1]

IEnumerator generates TSQL without where so it brings all records from server

## IQueryable Generates

SQL

```
SELECT
[Extent1].[ID] AS [ID],
[Extent1].[Name] AS [Name],
[Extent1].[Age] AS [Age]
FROM [dbo].[Employee] AS [Extent1]
WHERE 1 = [Extent1].[ID]
```



EventClass	TextData	RowCounts
SQL:BatchCompleted	declare @...	1
Audit Logout		
RPC:Completed	exec sp_reset_connection	
Audit Login	-- network protocol: LPC set quoted_identifier...	
RPC:Completed	declare @p1 nvarchar(64) set @p1=N'C.O.9.45' ...	
RPC:Completed	declare @p1 nvarchar(64) set @p1=N'C.O.9.45' ...	
SQL:BatchStarting	d...	
SQL:BatchCompleted	d...	1
SQL:BatchStarting	declare @...	
SQL:BatchCompleted	declare @...	1
Audit Login	-- network protocol: LPC set quoted_identifier...	
SQL:BatchStarting	SELECT [Extent1].[ID] AS [ID], [Extent1].[N...	
SQL:BatchCompleted	SELECT [Extent1].[ID] AS [ID], [Extent1].[N...	5
Audit Logout		
RPC:Completed	exec sp_reset_connection	
Audit Login	-- network protocol: LPC set quoted_identifier...	
SQL:BatchStarting	SELECT [Extent1].[ID] AS [ID], [Extent1].[N...	
SQL:BatchCompleted	SELECT [Extent1].[ID] AS [ID], [Extent1].[N...	1
Trace Pause		

IQueryable generates TSQL with where so it brings filtered records from server

```
SELECT
[Extent1].[ID] AS [ID],
[Extent1].[Name] AS [Name],
[Extent1].[Age] AS [Age]
FROM [dbo].[Employee] AS [Extent1]
WHERE 1 = [Extent1].[ID]
```

## ICollection:

ICollection inherits from IEnumerable. There is one difference:

You can find IEnumerable[ i ] --> Index Based

You can NOT find ICollection[ i ] --> Not Index Based

//IList {indexer and Modify} vs ICollection {randomly and Modify}

//Collection can not be instantiate from ICollection , so it should be instantiate from List

```
System.Collections.Generic.ICollection<string> strICollection = new List<string>();
```

```

strICollection.Add("Mahsa");
strICollection.Add("Hassankashi");
//Countable***
int ICollectionCount=strICollection.Count;
this.ListBoxICollection.DataSource = strICollection;
this.ListBoxICollection.DataBind();
System.Text.StringBuilder str = new System.Text.StringBuilder();
foreach (var item in strICollection)
{
    str.Append(" , " + item);
}
this.lblICollection.Text = str.ToString();
//IList***
System.Collections.Generic.IList<Employee> objIList = new List<Employee>();
objIList = (from m in ctx.Employees select m).ToList();
Employee obj = objIList.Where(i => i.Name == "Sara").FirstOrDefault();
int indexofSara= objIList.IndexOf(obj);
int cIList = objIList.Count;
//ICollection***
System.Collections.Generic.ICollection<Employee> objICollection = new List<Employee>();
objICollection = (from m in ctx.Employees select m).ToList();
Employee objIC = objICollection.Where(i => i.Name == "Sara").FirstOrDefault();
//You can not get index of object , if you clear comment from below code appears error
// int indexofSaraICollection = objIC.IndexOf(objIC);
int cICollection = objICollection.Count;

```

## Dictionary and IDictionary:

Dictionary and IDictionary both use the general term "Dictionary," although Hashtable does not: Dictionary TKey, TValue>. There is a minor distinction between them in that a dictionary will throw an exception if a certain key cannot be found, whereas a hashtable would

simply return null. Use IDictionary instead of Dictionary if you anticipate significant changes in the future.

```
//Dictionary can instantiate from Dictionary , Dictionary is similar to  
Hashtable,  
//Dictionary is GENERIC but Hashtable is NON GENERIC  
//Such Hashtable you can find object by its key  
System.Collections.Generic.Dictionary<int,  
string=""> objDictionary = new Dictionary<int, string="">();  
objDictionary.Add(1001, "Mahsa");  
objDictionary.Add(1002, "Hassankashi");  
objDictionary.Add(1003, "Cosmicverse");  
string str = objDictionary[1002];  
this.ListBoxDictionary.DataSource = objDictionary;  
this.ListBoxDictionary.DataBind();</int,></int,>
```

**SRIHARSHINI G**

**III-IT**

## Think Logically...,

### Problem 1:

Create a class ArrayListMain and in the main method get the names and store them in an ArrayList. After getting all the names, just display them in same order

#### Input format:

Number of names(N) in the first line as Integer

N names in separate lines

#### Output format:

Print the names

For answer refer Page No:38

### Problem 2:

Write the generic functions that can be called with different types of arguments based on the type of arguments passed to generic method, the compiler handles each method. Obtain an integer and double value and pass it to the function. Function Header: Static <T> void generic display (T element)

#### Input format:

The input consists of integer and double value

#### Output format:

The output prints the input values

Refer sample input and output for formatting specifications.

For answer refer Page No:39

### Problem 3:

Create a list of cost that has to be paid for halls and print only the even values using lambda expressions

#### Input format:

The first line of the input is the value of n

Next input is the cost separated by a space.

#### Output format:

The output prints the even values separated by a space.

For answer refer Page No:39

## **.NET Core nginx and Postgres with EF on an rPi**

Utilizing a rPi, creating an SSL-capable server in .NET Core WITHOUT ASP.NET, testing Postgres with EF, and using nginx.

This article begins by demonstrating how to set up rPi. The installation of PostgreSQL and DotNet Core will be covered after looking at Putty and WinSCP. Next, we'll look at how to connect to Postgres on the Raspberry Pi using C# and .NET Core. We'll then look at nginx, a very basic server, before examining how to run the server as a service.

### **Introduction:**

I've chosen to concentrate on the Internet of Things (IoT) this year, particularly the Raspberry PI (rPi) and Beaglebone Single Board Computers (SBCs). This will be my "personal year of IoT," I guess. My initial research is focused on determining whether one of these devices can be used as a straightforward web server (it can, of course), but also has the processing capability to host a SQL database and a "true" web application. I'm going to use a website I created for a customer a few years ago as a test case, but for the purposes of this blog, I only want to evaluate different technologies.

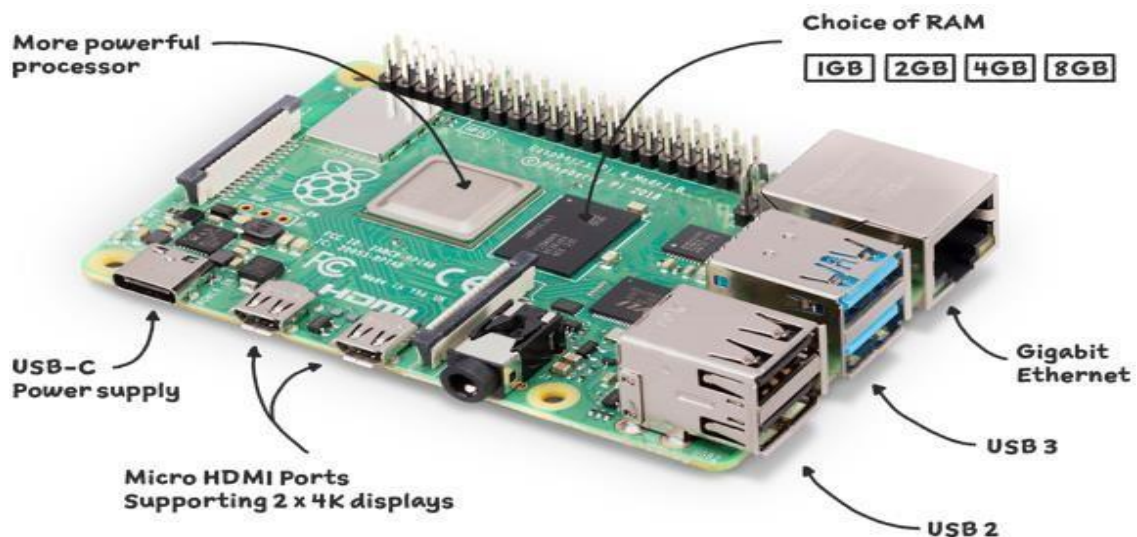


I should be able to determine the possibility of employing a very inexpensive SBC as a reliable web server for low-bandwidth purposes at the end of the day (not this article-day). I might experiment with some of the hardware features along the road. I have a personal history of



disliking MySQL, so I'm aiming to use PostgreSQL for my main goal and DotNet Core as the web server. There are three initial difficulties to be overcome:

- ✓ Postgres database import from a SQL Server database (a Docker image of SQL Server is not viable as it requires 2GB of RAM and the rPi only has 1GB RAM and the bigger issue is that it supports only x64 processors, not ARM processors.) Here is a Postgres Entity Framework Core supplier.
- ✓ ASP.NET is not used by the website I created for my customer, so I had to force .NET Core to utilise my server code.
- ✓ Learning how to use .NET Core and SSL certificates in the Linux environment.
- ✓ The next step entails setting up a reverse proxy server, which I do with nginx.



There will therefore be some difficulties that have nothing to do with SBCs and others that do!

### **Starting Out with the rPi:**

The KanaKit rPi needs to be put together and powered on first. Then, rather of using the micro SD card, I want to be able to boot the OS from a USB device.

### **OS Unpacking, Installation, and Loading:**

Opening the large package reveals the Raspberry Pi (rPi), a power supply, an HDMI cable, an SD card, a casing, and heat sinks inside:



After opening those boxes, we can access the hardware!

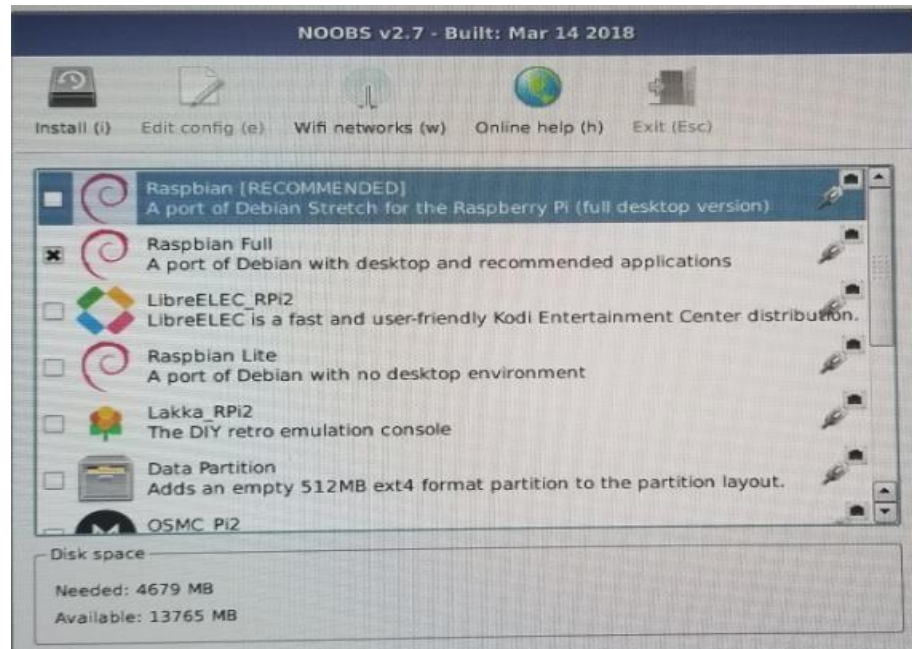


I refused to watch the video, but it took me at least 10 or 15 minutes to put the rPi in the case. It turns out that in order to properly position the SD card slot in the case's access hole, the rPi needs to sort of slide into place. After everything was put together (including the heat sinks):





NOOBS, or "New Out Of the Box Software," is pre-installed on the SD card and allows you to select the OS you want to use:



The installation process hung when I tried to install the advised Raspbian OS (the first checkbox) (I did wait several minutes.) I dreaded restarting the rPi, but when I did it cheerfully booted back into NOOBS. Raspbian Full, the second selection, installed without any issues. Great!

### **Examining the equipment and capabilities:**

The built-in WiFi on this model of the Raspberry Pi is one of its wonderful features, and configuring it was a breeze. Since I wanted to be able to work on this project outside of my workplace where I have cable connectivity, WiFi was a need, but I've never been able to get it working on the BeagleBone Black (my office is cluttered enough as it is with BeagleBone projects for another client.)

**ASHWIN KUMAR M**

**IV-IT**

## Using SQLite in C#/VB.Net

### Introduction:

The need for editing and displaying complicated data structures in everyday applications is becoming more and more prevalent due to recent improvements in memory and processing power of PC, server, and laptop computers. This article introduces the well-known and extensively used open source database SQLite in C#/.Net 101 fashion (note that the spelling is not SQLLite or SQL Light).

The SQLite database engine can be used as an in-memory engine or as a local file-based database engine and is compatible with a variety of operating systems (Android, IOS, Linux, Windows). Since the engine is implemented in a collection of DLLs that are referred to in a specific VS project, no additional Set-up is necessary. We are able to use an embedded database engine like SQLite.



- ✓ Having to create sophisticated structures from scratch, such as an additional index, or
- ✓ Needing more setup, maintenance, and security work to run a dedicated database server.

Among the uses of embedded databases are, but are not restricted to:

- ✓ preserving and retrieving data structures in the best possible manner (Application File Format).
- ✓ doing complex calculations instantly without the need for a separate server.

### SQLite Default Limitations:

Always closing a database after use is crucial since a possible hanging thread could prevent a later call to Open since a file-based database cannot be accessed by two threads at once (). The proper journal mode can be used to get past this default restriction, as we will see in the

section below on accessing a SQLite database from several threads. Foreign Key enforcement is by default disabled for backward compatibility reasons. The sections on enforcing foreign keys provide information on how to do it.

**Prerequisite:**

NuGet may be used to access the SQLite database engine. Simply start a fresh Visual Studio project (such as Console) and search for the System Data SQLite package. Install the package, then begin programming.



The binaries can also be downloaded manually from the following websites:

- ✓ <http://system.data.sqlite.org/index.html/doc/trunk/www/downloads.wiki> or
- ✓ <http://www.nuget.org/packages/System.Data.SQLite/> (rename nuget file to zip and unzip)

### Background:

Following the publication of my Advanced WPF TreeView articles, I received the following comments:

## How can I save or get data based on TreeView from/to a file?

I'm taking this detour to describe the fundamentals of SQLite in order to respond to this using SQLite in an applied manner, and I'll build on it in a later article by describing how to store tree-structured data in a relational database engine (see also XML solution over here).

### Additional Tools:

Typically, a relational database system will have a text-based SQL query application (e.g.: SQL Management Studio or VS for SQL Server). There are numerous such client query programmes in the SQLite environment. A SQLite/SQL Server Compact Toolbox Plug-in for Visual Studio, for instance, can be used to state SQL queries or check the database's current Data Definition.

Contrarily, I am a major proponent of open source, thus for this and other reasons I use the SQLite Manager Plug-in for FireFox, which is accessible on all systems and does not alter my Visual Studio settings.

### Using the Code:

Simple WPF projects with text output make up the tutorial's sample applications. The file MainWindow.xaml.cs contains the code (unless stated differently).



### A 'Hello World' SQLite Database:

1. Download 00\_SQLite\_tut.zip
2. Download 01\_SQLite.zip

Although the SQLite code in the sample projects is fairly self-explanatory, those unfamiliar with working with relational database systems may find it confusing. Here are some examples that clarify the fundamentals:

### Creating a database (file):

You can use the following snippet to create a SQLite database file:

```
// create a new database connection:  
SQLiteConnection sqlite_conn = new SQLiteConnection("Data  
Source=database.sqlite;Version=3;");  
  
// open the connection:  
SQLiteCommand sqlite_conn.Open();
```

A connection string is passed as a function Object() { [native code] } parameter when creating a SQLiteConnection object in the first line. This command will generate a database.sqlite file in the current directory's Debug/bin or Release/bin folder. Version 3 of SQLite is the least necessary version.

```
// create a new database connection:
SQLiteConnection sqlite_conn =
    new SQLiteConnection("Data Source=:memory::Version=3;");
// open the connection:
SQLiteCommand sqlite_conn.Open();
```

An in-memory SQLite database is created by the previous command. The connection is severed when an instance of an in-memory database expires, making each instance distinct.

### **Create a Table in the SQLite File:**

The following statement creates a table and a new file-based database. (The SQLite system does not (by default) recreate a table. If the statement is executed twice, a corresponding exception should be thrown):

```
// create a new database connection:
SQLiteConnection sqlite_conn =
    new SQLiteConnection("Data Source=database.sqlite;Version=3;");
// open the connection:
SQLiteCommand sqlite_conn.Open();
SQLiteCommand sqlite_cmd = sqlite_conn.CreateCommand();
// Let the SQLiteCommand object know our SQL-Query:
sqlite_cmd.CommandText = "CREATE TABLE test (id integer primary key, text
varchar(100));";
// Now lets execute the SQL ;-)
sqlite_cmd.ExecuteNonQuery();
```

Almost every SQL operation on the SQLite system can be carried out using the last two lines. While other methods, like ExecuteReader(), can be used to get (extended) results of that query defined in the CommandText field, you typically end up utilising the ExecuteNonQuery() method to execute operations on the data dictionary (create, drop, etc.). To construct a table only

once, use the create table statement below. Otherwise, each time you select or insert data into the table, it is reused (see 01 SQLite tut.zip).

```
sqlite_cmd.CommandText =  
    @"CREATE TABLE IF NOT EXISTS  
    [Mytable] (  
    [Id]    INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,  
    [NAME] NVARCHAR(2048) NULL)";
```

### **Insert a Record into a Table:**

This "Hello World" example inserts 1 record with 2 values into a table that already exists.

```
SQLiteConnection sqlite_conn =  
    new SQLiteConnection("Data Source=:memory::;Version=3;New=True;");  
SQLiteCommand sqlite_conn.Open();  
SQLiteCommand sqlite_cmd = sqlite_conn.CreateCommand();  
sqlite_cmd.CommandText = "INSERT INTO test (id, text) VALUES (1, 'Hello World');";  
sqlite_cmd.ExecuteNonQuery();
```

### **Read a Record from a Table:**

Here is a 'Hello World' example that reads records with 2 values from an existing table:

```
SQLiteConnection sqlite_conn;        // Database Connection Object  
SQLiteCommand sqlite_cmd;            // Database Command Object  
SQLiteDataReader sqlite_datareader;  // Data Reader Object  
sqlite_conn.Open();  
sqlite_cmd = sqlite_conn.CreateCommand();  
sqlite_cmd.CommandText = "SELECT * FROM test";  
sqlite_datareader = sqlite_cmd.ExecuteReader();  
// The SQLiteDataReader allows us to run through each row per loop  
while (sqlite_datareader.Read()) // Read() returns true if there is still a result line to read  
{  
    // Print out the content of the text field:  
    // System.Console.WriteLine("DEBUG Output: '" + sqlite_datareader["text"] + "'");  
}
```

```

object idReader = sqlite_datareader.GetValue(0);
string textReader = sqlite_datareader.GetString(1);
OutputTextBox.Text += idReader + " " + textReader + " " + "\n";
}

```

- ✓ The while loop in the aforementioned code snippet continues continuously until the query produces no more result data, or it never runs if there are no results that satisfy the query.
- ✓ The GetValue(1) method, as demonstrated previously, yields a .Net object value that can be used to reflect into other types. The identical outcome can also be obtained using the field index-based syntax: `sqlite_datareader["text"]`.
- ✓ If you are aware of the target, you can also utilise alternate obtain value methods on the `SQLiteDataReader` object, like `GetString(1)` or `GetInt32(1)`. Based on the SQLite data, the .Net data-type should be used:

```

bool GetBoolean(int i);
byte GetByte(int i);
char GetChar(int i);
DateTime GetDateTime(int i);
decimal GetDecimal(int i);
double GetDouble(int i);
float GetFloat(int i);
Guid GetGuid(int i);
short GetInt16(int i);
int GetInt32(int i);
long GetInt64(int i);

```

Since you just built it, you either already know the correct data type for a `SQLiteDataReader` column, or you can use the following technique to deterministically discover the correct data type of a given column:

```

// Retrieves the name of the back-end data-type of the column
string GetDataTypeName(int i);
// Returns the .NET type of a given column

```

```
Type GetFieldType(int i);  
// Retrieves the name of the column  
string GetName(int i);  
// Retrieves the i of a column, given its name  
int GetOrdinal(string name);  
// Returns the original name of the specified column.  
string GetOriginalName(int i);
```

**MONISHA E**

**IV-IT**



## Think Logically solutions

### Problem 1:

```
import java.util.*;
public class ArrayListMain1 {
    public static void main(String args[]){
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        ArrayList<String> list=new ArrayList<String>(n);
        String name;
        for (int i = 0; i <= n; i++)
        {
            list.add(sc.nextLine());
        }
        Iterator itr=list.iterator();
        while(itr.hasNext())
        {
            System.out.println(itr.next());
        }
    }
}
```

### Sample Input:

```
6
Raji
Jai
Saro
Vijay
Ravi
Amudha
```

**Problem 2:**

```
import java.util.*;
class Q3 {
    static <T> void genericDisplay(T element)
    {
        System.out.println(element.getClass().getName()
            + " = " + element);
    }
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        int val=sc.nextInt();
        double d=sc.nextDouble();
        genericDisplay(val);
        genericDisplay(d)
        //genericDisplay(1.0);
    }
}
```

**Sample Output:**

```
10 5.23
java.lang.integer=10
java.lang.double=5.23
```

**Problem 3:**

```
import java.util.ArrayList;
import java.util.*;
class Test
{
    public static void main(String args[])
    {
        Scanner sc=new Scanner (System.in);
```

```
int nn=sc. nextInt();
    ArrayList<Integer> arrL = new ArrayList<Integer>();
for(int i=0;i<nn;i++)
{
//arrL[i]=sc.nextInt()
    arrL.add(sc.nextInt());

}
//arrL.forEach(n -> System.out.println(n));
arrL.forEach(n -> { if (n%2 == 0) System.out.print(n+" "); });
}
}
```

**Sample Output:**

3

1 2 3

2

# EDITORIAL MEMBER

## EDITORIAL FACULTY MEMBER

Ms. T. YAWANIKHA AP/IT

## EDITORIAL STUDENT MEMBERS

Anandharaj G – ( III IT )

Bhuvana K T – ( III IT )

Divya K – ( III IT )

Hari govind S – ( III IT )

**GET IN TOUCH NOW!**



**Web:** [www.karpagamtech.ac.in](http://www.karpagamtech.ac.in)

**Address:** L&T Bypass Road,  
Bodipalayam post,  
Coimbatore 641 105.