Free Query Tool - tool for enhanced querying SQL databases

Department of Software Engineering

Supervisor: RNDr. Michal Kopecký, Ph.D.
Program: Computer science, programming

2008
First of all I would like to thank to the supervisor of my bachelor thesis RNDr. Michal Kopecký, Ph.D. (Charles university in Prague) for his valuable comments and help and for his neverending patience. I would like to thank also to professor Terence Parr (University of San Francisco) for such great tool that ANTLR is. Thank you both.

I declare I wrote the Bachelor’s thesis myself, using only the referenced source. I approve lending of this thesis.

In Prague 8. 8. 2008

Martin Šuška
# Index

1. Introduction ........................................................................................................... 6
  1.1 Motivation ........................................................................................................ 6
  1.2 Goal .................................................................................................................. 6
  1.3 Thesis structure ............................................................................................... 6

2. Analysis .................................................................................................................. 7
  2.1 Syntax highlighting ......................................................................................... 8
  2.2 Database metadata gathering ........................................................................ 8
  2.3 Code templates ............................................................................................... 9

3. Design and implementation ................................................................................. 10
  3.1 FQT architecture overview ............................................................................ 11
    3.1.1 Common component ............................................................................... 12
    3.1.2 FQT component .................................................................................... 12
    3.1.3 FQT Oracle module ............................................................................... 12
  3.2 Packages and classes ...................................................................................... 12
    3.2.1 net.freequerytool.core ......................................................................... 12
      Main ............................................................................................................ 13
      ActionController ......................................................................................... 13
      FQTConnection ......................................................................................... 13
      Shortcut ...................................................................................................... 13
    3.2.1.1 net.freequerytool.core.background
      ClassProcess ............................................................................................. 13
      ExecuteQueryProcess ............................................................................... 13
    3.2.1.2 net.freequerytool.core.dbmodules
      ModuleFactory ........................................................................................... 13
      ModuleConfigurationException ................................................................ 14
    3.2.1.3 net.freequerytool.core.settings
      GlobalSettings ............................................................................................ 14
  3.2.2 net.freequerytool.gui .............................................................................. 14
  3.2.3 net.freequerytool.common ....................................................................... 14
  3.2.4 net.freequerytool.modules
    IModule ....................................................................................................... 14
    IParser ......................................................................................................... 15
    ICommandParser .......................................................................................... 15
    IParseModel ............................................................................................... 15

4. User manual ........................................................................................................... 16
  4.1 Running application ...................................................................................... 16
  4.2 Log in to database ......................................................................................... 17
    4.3.1 First query ............................................................................................. 18
    4.3.2 Code completing .................................................................................... 19
    4.3.3 Code templates ..................................................................................... 19
    4.3.4 Backdoors ............................................................................................ 19
      4.3.4.1 Force execution .............................................................................. 19
      4.3.4.2 Force highlighting ......................................................................... 20

5. Conclusion ............................................................................................................ 21
  5.1 Comparison with other tools ......................................................................... 21
    5.1.1 SQuirreL ............................................................................................... 21
    5.1.2 SQLTools ............................................................................................... 22
    5.1.3 SQL developer ...................................................................................... 22
5.1.4 Comparison .............................................................................................................. 22
5.1.4.1 Highlighting ......................................................................................................... 22
5.1.4.2 Export as insert statements .................................................................................. 23
5.2 Future plans .................................................................................................................... 23
CD Content .......................................................................................................................... 25
Bibliography ........................................................................................................................ 26
Názov práce: Free Query Tool – nástroj pro dotazování do SQL databází  
Autor: Martin Šuška  
Katedra (ústav): Katedra softwarového inženýrství  
Vedoucí bakalářské práce: RNDr. Michal Kopecký, Ph.D.  
e-mail vedoucího: kopecky@ksi.mff.cuni.cz  
Abstrakt: Cielem práce je navrhnut a implementovat SQL konzoly. Nástroj by měl využívat informace o schématu databáze k zvýrazňování syntaxe, pomáhat s doplňováním jmen objektů, uvádět potřebné parametry funkci a jejich typy. Systém by měl být zaměřen na databázi Oracle s minimalizováním závislosti ke konkrétní databázi k umožnění pozdějšího rozšíření pro jiné databáze.  
Klíčová slova: FQT, SQL, dotazování, databáze, Oracle

Title: Free Query Tool - tool for enhanced querying SQL databases  
Author: Martin Šuška  
Department: Department of Software Engineering  
Supervisor: RNDr. Michal Kopecký, Ph.D.  
Supervisor's e-mail address: kopecky@ksi.mff.cuni.cz  
Abstract: The aim of this thesis is to design and implement an SQL console. This tool should utilize information about the schema for syntax highlighting, database object name hinting, indicating needed parameters and their data types. The system should be targeted primarily for Oracle. Try to minimize the dependency on the particular database system to simplify future portation to other systems.  
Keywords: FQT, SQL, querying, database, Oracle
1. Introduction

1.1 Motivation
SQL developers, while developing their queries, spend a lot of time searching correct table, view, columns (all together generally DB object) name spelling, they search quite often relations in database, mainly when their schemas are greater than scott\(^1\), they are also looking for function arguments and other constraints, therefore it's a good idea to create SQL console, that enables user to complete such things automatically. I'm assuming that SQL developer knows at least basic syntax of SQL commands and therefore we need to help him mainly with object names completing.

1.2 Goal
In this thesis I would like to answer questions about making relation database querying more effective. I want to implement the tool that helps user be more effective in query creation. I would describe the ways, how to accomplish this. These methods will be described in detail later in this thesis and it will be shown how FQT (I will use this abbreviation for Free Query Tool) implements these methods. At the end there will be some comparison with other freely available tools for database querying.

1.3 Thesis structure
In following chapters I'm describing:

- **analysis** – chapter is finding answers to questions what can help user to become more effective
- **design** – is describing how I want to realize these mechanisms
- **implementation** – covers actual implementation, describes structure and main classes of the project
- **user manual** – represents help for user about how to install, run and use FQT tool
- **conclusion** – I will summarize implementation results, comparing with other tools as well

---

\(^1\) scott is sample schema in Oracle database containing by default only four tables
2. Analysis

There is a lot of questions that needs to be answered to create helpful tool for database querying, for example: „What the user really needs? What can help user to create queries quicker? What the user spends his time on? What are the possibilities to help user be more effective?“

What is the purpose when user starts to write query? He wants to see data stored in database, these data can be aggregated, there could be used functions modifying these data somehow, but in fact user wants to see result returned for his query from database. We do not know what kind of data will user need, therefore exists SQL. So what can we do to help user with query writing? Users spend a lot of time with checking which columns creates primary of foreign keys in table. They make typing errors in commands (queries), table/view/column names or they simply do not remember the names (what is understandable, if there is big database or user works with a more databases). Why the tool that user is working with cannot help him with it? It can and we will discuss the possibilities here.

Does the user need visualization of the query? This feature could be fine when creating project documentation. When talking about effective query writing definitely not. Of course you cannot have typing error in something you didn’t wrote, but there are limitations comparing to possibilities provided by database – if there is query that is somehow interesting (logically difficult), there is only small probability that visualization of such query will be better. Consider situation that I have employee table (emp) which contains hire date and I want to first five users that are in company the longest time. Such query will be:

```sql
select * from ( 
  select * 
  from emp
  order by hiredate
) where rownum <= 5
```

I cannot imagine how visualization of such query can be easier to create instead of writing it and this is not complex query at all.

Does the user require feature that he can create whole query using mouse only? I think no, using mouse in applications is great thing, but mouse in not helpless when writing is needed. In fact I do not think that mouse usage is quicker than typing also in case that everything in application supports drag’n’drop. Some simple operations like select all from some table could be quicker, but generally this is not the truth, especially conditions specifying is difficult to realize with mouse neither with mouse, using in one hand, and keyboard, using with another one. On the other hand possibility of
mouse using for query creation could be very helpful when user is learning SQL and he is discovering the possibilities of the language.

I was talking about some features that really exists in some tools, but they do not help to reach the goal of making querying easier – more effective. The features that I found useful are: syntax highlighting, database metadata gathering, code templates.

2.1 Syntax highlighting
Syntax highlighting is a technique where some parts of text are pointed out to create more readable code, to warn user about some syntax errors and so on. What highlighting means? Highlighting changes text style of some significant code, using bold text, underlined text, text with changed text color etc. The highlighting can be very helpful in two situations:

- in case of the complex query the error can be shown on the exact place in the query, while the error message from the database is not much meaningful sometimes.
  
  consider following query:

  ```
  select *
  from emp
  and sal > 1000;
  ```

  Oracle database produces following error:

  ORA-00933: SQL command not properly ended

  I'm confident that user looks at the end of query, he sees ';' and is confused that his query is properly ended. Real problem in query above is that it's missing WHERE keyword.

- or when the code is not highlighted when you expect it, you can solve the problem immediately.

  There are also some frequent errors that users do - such as typing errors when user writes SEELECT instead of SELECT or FORM instead of FROM. In these cases when user knows that keywords are highlighted standardly and highlighting is missing, he starts solving the problem even before executing the query.

2.2 Database metadata gathering
The purpose of databases is mainly in storing data and support for their manipulation, but there are often other informations in databases. Informations about stored data – so called metadata. This information typically contains a list of all objects in the database (tables, views, …), it contains also
further information about objects, for example table metadata contains information about columns in the table, its constraints and so on. Query tool can gather and show such an information to make querying more effective.

2.3 Code templates
Code templates are code snippets generated automatically when user invokes such functionality. In SQL what means Structured Query Language there are great opportunities to use templates. While query has structure, this structure can be created using those templates.
3. Design and implementation

In previous chapter we discussed features that could make querying more effective. This chapter describes possibilities of accomplishing those requirements.

Basic idea of implementation is to split application to two parts – the database dependent and database independent one. There is a lot of reasons why to do so. One example for all - database dependent could be the way how to retrieve informations about tables in database. In Oracle database there are views named *_tables*, that contains informations about tables, in MySQL database exists show command. Also relevant reason is possibility to enable other modules implementation in future.

**Code templates** are the easiest to implement, we have to define the action when the code template have to generate some code. Let this action be some keyboard shortcut, because the user do not need to stop writing, he just need to memorize defined or set up own shortcuts for offered actions.

**Database metadata gathering** is database dependent operation, so we will create interface that module implementor have to realize to enable this feature.

**Syntax highlighting** is the most difficult feature from previously described ones. For realization of this feature the tool have to „understand“ SQL. It is also database dependent. Someone could object to while SQL means structured query language there is well defined structure of SQL commands, but reality is not so easy. Of course there is a structure, for example select command have to contain SELECT word, list of items to select, FROM word and list of objects to select from (these objects could be somehow joined or filtered, it's not important here). There do not need to be only table, but view as well. Oracle database enables inner select here and even function, Oracle online documentation for Oracle 11g [Oracle11g] says for table collection expression:

> „The table_collection_expression lets you inform Oracle that the value of collection_expression should be treated as a table for purposes of query and DML operations. The collection_expression can be a subquery, a column, a function, or a collection constructor.“

Oracle also enables user to switch having and group by parts in select command and this is not enabled in MySQL (tested version 5.0.38), so I decided that responsibility for command parsing takes database module.

Module have to implement parser. Parser is kind of really complex class (parser for Oracle SQL

---

2 star should be replaced with user, dba, all depending on user rights
have more than 20 000 lines of code) so I decided to use some freely available tool for parser generation. I was considering two possibilities JavaCC [JavaCC] and ANTLR[ANTLR]. I heard about ANTLR before, there is more\(^3\) WWW pages about ANTLR and there exists ANTLRWorks [aw] so you can familiarize and work with ANTLR quicker.

For logging in whole application is used log4j library. Java have own logging framework, but it misses a lot of features comparing to log4j.

For implementation I decided to use Java 5 (java 1.5.0_15), it's older version of java (actual when writing this thesis is java 1.6.0_07), this is because I think more users use older version, on the other hand features like generics, enumeration types and annotations came to java in version 5, so I stayed with Java 5 and didn't used older version. Implementation of background processes is based on SwingWorker [sw] which is part of Java 6, so I used Java 5 backport.

For connecting and executing commands I'm using jdb library for Java 5 (for Oracle it's ojdbc5.jar).

### 3.1 FQT architecture overview

FQT contains logically two parts (components)

- DB modules that communicate with databases, are dependent on details in the database query language and the client-server communication protocol. Its role is to encapsulate all database dependent code and provide database independent interfaces for other modules of the application.

- Main FQT code.

As there are some dependencies between modules and FQT components, I decided to create common subsystem to avoid cycle dependencies and to separate the shared code. Figure 1 shows components in FQT.

---

\(^3\) search results using Google are in ratio 250 000:650 000 – JavaCC:ANTLR
3.1.1 Common component
It contains interfaces and classes that are used in both FQT and DB modules. Common functionality used in both parts is logging. There are interfaces which are used to send messages (call methods) between these parts. For example module is responsible for parsing input text, but FQT component is responsible for highlighting of parsed command, because this is database independent. Concrete details will be discussed later in this chapter.

3.1.2 FQT component
As mentioned above FQT component is responsible for example for highlighting, in general it's responsible for database independent code, like application GUI (event handling belongs here too), configuration and settings.

3.1.3 FQT Oracle module
Represents module for Oracle DB querying. There are just dependent things like lexer, parser, module class.

3.2 Packages and classes
Application contains four main packages. These packages do not follow logical separation strictly, there are some overlaps, but this is natural. Situation is described on figure 2.

3.2.1 net.freequerytool.core
Package itself contains important classes and also other subpackages, shown at figure 3.
Main
This is application entry point, class checks command line attributes and starts application.

ActionController
Handles events in application such as button clicks, mouse clicks, there are some exceptions when this class is not used.

FQTConnection
Class wraps everything that has something with database connection. In terms of FQT connection represents one connection to database and it could have more sessions and one reference to DB objects. FQT enables creation of more connections to one or more (different/same) databases.

Shortcut
Represents keyboard shortcut and it should cover basic information about pressed button, like modifiers and pressed key. It's primary designed as key for map of shortcuts for SQL Area.

It's implemented as fond, because number of shortcuts is limited in real life.

3.2.1.1 net.freequerytool.core.bgthreads
Package contains classes that implements background processes. These classes extends SwingWorker class, see for example [sw] for implementation principles.

ConnectProcess
Class implements login process to database. When user fills in database information such as host, port, user and password this process creates connection using these informations.

ExecuteQueryProcess
Executes queries in background to prevent application from freezing.

3.2.1.2 net.freequerytool.core.dbmodules
Package contains classes that are related to module configuration mechanism, which is following: there is module configuration file – module.properties that contains information about module. Then Java reflection is used to create instances of classes that are described here.

ModuleFactory
Manages whole module configuration.
ModuleConfigurationException
This is runtime exception, because when modules are configured properly there is no way how this exception can occur.

3.2.1.3 net.freequerytool.core.settings

GlobalSettings
Contains configuration such as module property file name (module.properties), the names of directories such lib/, prop/, metadata...

3.2.2 net.freequerytool.gui
This package contains all classes that represents GUI (for example MainWindow class, PasswordDialog, ConnectionDialog and other) or help to work with such classes (for example GUIHelper which offers function to centralize window). Almost all classes are really simple, just SQLArea class is more complex, so it has own package and other related classes are there too. When user press some shortcut new command class is created and executed, command classes are in own package under sqlarea package, situation is depicted on figure.

3.2.3 net.freequery.common
This package at the moment contains only the LogHelper class, that simplifies logging in application.

3.2.4 net.freequerytool.modules
Package contains all interfaces to enable communication between FQT and module components, these interface are in FQT common component and concrete module implementation belong to module component.
**IModule**
Interface enables FQT component to use module. Methods return module dependent objects and therefore return types are also interfaces defined in FQT Common component, there is only one exception – method `getLexer()` returns `Lexer` class reference, however it is abstract class, it is not interface defined in FQT but it's not problem, because for now I suppose that ANTLR [antlr] is used syntax highlighting.

**IParser**
Represents the minimal functionality required from parser. This interface hides implementation details and leaves only functionality relevant for FQT component – needed when initializing module.

**ICommandParser**
Instances of this class handle command parsing. It used widely in FQT component.

**IParsedCommand**
Indirectly offers methods for command highlighting. When command is parsed, command parts are categorized and highlighted later, examples of such categories are strings, function names, keywords and others. It offers other methods to modify parsed command, some of them are used in FQT component for example when parsing command from `SQLArea`, parser gets only the command(s) chanded by user, not whole `SQLArea` content. When command is parsed offset of such command is set to 0, but if it wasn't first command in `SQLArea` offset is greater than 0 and should be moved – `move()` method. Interface also offers methods for command type setting and retrieving, what is used in module and also in FQT component.

FQT Oracle module contains other packages as well. These packages are

`net.freequery.modules.lexers` - contains Lexer related classes (for Oracle module there is just Lexer class generated using ANTLR [antlr]) and `net.freequery.modules.parsers`
- contains Parser related classes, in FQT Oracle module there is `OracleCommand` class
- implementation of `IParsedCommand`. 


4. User manual

4.1 Running application

Application is available in zipped archive fqt.zip, on CD is also unzipped version (both described in CD content chapter). Running FQT cannot be easier, just run start_FQT.sh on unix-like systems:

```
$ ./start_FQT.sh
```

or run start_FQT.bat on Windows

```
> start_FQT.bat
```

in few seconds FQT main window should be displayed.

![FQT main window](image)

*Figure 5: FQT main window*
4.2 Log in to database

As shown on main windows screen shot (Figure 5), there is help page that displaying information how to log in to the database. Choose Tools > Connect from menu or simply press keyboard shortcut $Alt + C$.

![Figure 6: Tools > Connect](image)

When invoked Connection dialog is displayed and enables user to insert connection informations or simply log in with saved connection. Also when connection is stored FQT does not remember the passwords, there are other tools for this for example mypasswordsafe or password-gorilla, the second one is available for windows too.

![Figure 7: Connection dialog](image)

Figure 7 shows connection dialog. On the right side are fields to add connection, when new connection is created, it is added to table. To connect using saved connection double click on row in saved connections table or select the connection and press connect button. 4.3 Query writing
4.3.1 First query

When user is connected to database SQL Area and result panel appear instead of welcome page.

![Figure 8: SQL Area and result panel](image)

User can start writing the queries. Queries are highlighted immediately. Shortcut for query execution is `CTRL + Enter`.

![Figure 9: query and statement result](image)
4.3.2 Code completing
Code completing is offered when user press Ctrl + Space shortcut.

Figure 10: code completing

4.3.3 Code templates
Code templates are invoked when keyboard shortcut is pressed. For select template press Alt + S, Figure 11 shows generated code.

Figure 11: select template

4.3.4 Backdoors
The golden programmer rule says: „Every time when you change the code, you added some error into it“. This is often true and that's why there is backdoor in FQT. FQT do not want to pretend it's perfect as I tried to find and remove all bugs to make it bug free, there is big probability there are some bugs I missed, because there is not a lot of users now.

How the backdoor in FQT works? There are two buttons in application menu that force execution of some actions.

Figure 12: backdoors

4.3.4.1 Force execution
This feature enables user to run query FQT did not understand. There is some probability that the query which the user wrote is difficult for FQT to recognize – there could be error in code of course, there could be some feature that is not in Oracle 11g or something is simply missing in
parser (FQT Oracle module knows just the subset of all possibilities that Oracle enables), FQT query rewriting.

In such cases there is possibility for user to force execution of the command he has written.

4.3.4.2 Force highlighting
As I wrote about my experience with SQuirreL that it turned off highlighting without obvious reason I decided to enable user to force highlighting of the text he has in SQL area. Maybe this feature wont be used ever (and that is good for me as programmer of the highlighting), but it is easier to enable such action instead of searching for all scenarios that user can do, of course basic scenarios are covered, but possibilities of copy'n'paste or whatever else is not predictable
5. Conclusion

FQT achieved goal of the software that wanted to do the things correctly instead of the a lot of things that are not written well. Tool is far from considered to be complete, mainly settings. But this is not the main goal of the project and this tool will be extended in the future. Tool introduced mechanisms such as own metadata and templates for generating code, it has got (Oracle module) own sql parser, that works correctly, better in other tools (see comparison).

I hope there will be a lot of users that admin FQT as usefull tool.

5.1 Comparison with other tools

There exists a lot of tools for querying relational databases from database providers, as well as from third parties . I will concern on freely available tools such as: SQuirreL, SQLTools, SQL Developer, as I have many personal opinions with them.

5.1.1 SQuirreL

SQuirreL [SQuirreL] is released under GNU Lesser General Public License [LGPL]. It is written in Java programming language [JSE], so it's platform independent. I used it on Windows XP and many times I experienced the situation when the application switched off the highlighting without any reason, I also had often problems with its graphical user interface. Sometimes it simply overlaped some program windows. I'm Java developer and still I do not know any reason for this behaviour. I hadn't such problems with any other Java application before.

SQuirreL, comparing to FQT, has support for more databases. There is currently only Oracle 11g module available for FQT. SQuirreL has also worse object presentation – DB objects are organized in list and this list contains objects for all schemas in database, I think user is typically interested in just one and FQT therefore use combobox for schema selection what is space sparing. Great feature in SQuirreL is its limitation of result set size (when you are selecting something), on the other hand it's the feature I didn't like the most, because when you are updating something the limitation is applied too, what is weird. Consider following situation: you are updating some records in table and when you forgot to switch off the limitation, it updates only the first $n$ records, where $n$ is the limit. Problem is, that this update was not performed atomically and so it could be very difficult to revert the changes. For example when there is some calendar present in the database and you want to move all appointments in this calendar one month forward and the database modifies the table partially, it could be impossible to find out which records were updated.
Requires only jdbc driver (JRE too of course).

I was testing version 2.6.6 released on 2008-06-13.

5.1.2 SQLTools
The SQLTools [SQLTools] provides very user-friendly feature – drag'n'drop possibilities. You can drag'n'drop values from result set to SQL area, from object list, as well as from object viewer.
Small problem I experienced with drag'n'drop feature was in mixing letter case – from result table it uses upper case (column names), from object viewer it uses lower case (and in viewer shown as upper case), in version I was testing object was not working (not sure if it's 11g bug, I used it before with 10gR2 and there was no problem).

Dissadvantages of SQLTools are the lack of portability – it's available only for Windows OS – and impossibility to cancel long running tasks. However, there is such a possibility on the dialog box shown when the query takes a long time, but it seems it works strangely as though it finishes the query (you are waiting) and then simply throws away the results (you waited and have no result).
Requires Oracle instanet client - OCI (Oracle Call Interface).
I was testing last stable release 1.4.2 stable RC3 released on 2006-11-1

5.1.3 SQL developer
SQL Developer is a tool developed by Oracle. It's written in Java, so it's platform independent. I do not know this tool so good as the others, but I choosed it, because it's developed by Database vendor and it could be great for comparison.

Requires JDK (really not only JRE), jdbc contains in archive.
I was testing version 1.5.1 (2008-06-09)

5.1.4 Comparison
In this chapter I would like to compare some functionalities of these tools. It'd not study that compares all functionalities, this is not in the scope of this thesis I would like to refer about features that woks correctly in FQT comparing with other tools.

5.1.4.1 Highlighting

```
select avg(sal) avg from emp
```
All tools except FQT interprets the query incorrectly, in meaning of highlighting. SQLTools and SQL developer highlighted alias as function, SQuirreL highlighted alias as error, probably expecting parenthesis and arguments.

5.1.4.2 Export as insert statements
Exporting selected data in interesting feature, it's not difficult to implement such functionality surprising I do not know tool where this feature works.

SQLTool does not support this feature.

SQL developer and SQuirreL generates incorrect output, but surprisingly the same, maybe they both call some function that is not working well.

Finally FQT generates the correct output.

5.2 Future plans
There is a lot of things that could be improved in FQT or simply missing at the moment:

- there are other databases, not only Oracle, for which modules could be implemented.
- user setting could be extended, but it depends on user what will they request.
• FQT could be more robust. Property files are freely editable and there is not implemented mechanism to check before starting using them.

• as I mentioned before FQT is not storing passwords in current version.
CD Content

\ 
|-- FQT
  |-- lang
  |-- lib
    |-- FQT.jar
    |-- FQT_Oracle_module.jar
    |-- FQT_common.jar
    |-- antlr-3.0.1.jar
    |-- log4j-1.2.15.jar
    `-- swing-worker-1.1.jar
|-- licenses
    |-- ANTLR-BSD-License.txt
    |-- Apache-License-2.0.txt
    |-- gpl-3.0.txt
    `-- lgpl-3.0.txt
|-- metadata
|-- props
    |-- log4j.properties
    `-- module.properties
|-- start_FQT.bat
    `-- start_FQT.sh
|-- JavaDoc
  |-- FQT
    |-- FQTCommon
      `-- FQTOracleModule
|-- Thesis
  `-- fqt.zip

CD structure
Bibliography


[JavaDoc]  http://java.sun.com/j2se/1.5.0/docs/api/


[LGPL]  http://www.gnu.org/licenses/licenses.html#LGPL


[Parr07]  Terence Parr. The definitive ANTLR reference


