Getting Started With

Hands-on exercises included

Open Source Development

Ideal for application developers and administrators

by:
Rachna Kapur
Mario Briggs
Tapas Saha
Ulisses Costa
Pedro Carvalho
Raul F. Chong
Peter Kohlmann

DB2 ON CAMPUS BOOK SERIES

GETTING STARTED WITH

Open source development

A book for the community by the community

Rachna Kapur, Mario Briggs, Tapas Saha, Ulisses Costa,
Pedro Carvalho, Raul F. Chong, Peter Kohlmann

FIRST EDITION

First Edition (July 2010)

IBM Canada
8200 Warden Avenue
Markham, ON
L6G 1C7
Canada

Notices

Trademarks

Table of Contents

Preface

Keeping your skills current in today's world is becoming increasingly challenging. There are too many new technologies being developed, and little time to learn them all. The DB2® on Campus Book Series has been developed to minimize the time and effort required to learn many of these new technologies.

Who should read this book?

This book is a good starting point for beginners to the open source world. It is specially written to equip students, and open source enthusiasts with the norms and best practices of open source. You should read this book if you want to:

- Educate yourself on the objectives of open source
- Understand open source software licensing requirements
- Get an introduction to the norms followed in the open source world
- Join the open source movement and begin contributing.

How is this book structured?

The first chapters of this book discuss the history of open source software development and its licensing requirements. It then talks about how organizations use open source as their business model. Chapter 4 introduces the reader to the tools used in the development of an open source project. Chapters 5 and 6 take the reader into more details about how to contribute to an existing open source project. Chapter 7 provides a case study where you practice contributing to an open source project. Chapter 8 goes a bit deeper describing the ***Technology Explorer for IBM DB2***, an open source project hosted at sourceForge.net; it also summarizes and revisits some of the concepts discussed in the previous chapters.

Exercises are provided with most chapters. There are also review questions in each chapter to help you learn the material; answers to review questions are included in *Appendix A*.

A book for the community

This book was created by the community; a community consisting of university professors, students, and professionals (including IBM employees). The online version of this book is released to the community at no-charge. Numerous members of the community from around the world have participated in developing this book, which will also be translated to several languages by the community. If you would like to provide feedback, contribute new material, improve existing material, or help with translating this book to another language, please send an email of your planned contribution to db2univ@ca.ibm.com with the subject “Getting started with open source development book feedback”

Conventions

Many examples of commands, SQL statements, and code are included throughout the book. Specific keywords are written in uppercase bold. For example: A **`NULL`** value represents an unknown state. Commands are shown in lowercase bold. For example: The **`dir`** command lists all files and subdirectories on Windows®. SQL statements are shown in upper case bold. For example: Use the **`SELECT`** statement to retrieve information from a table.

Object names used in our examples are shown in bold italics. For example: The ***`flights`*** table has five columns.

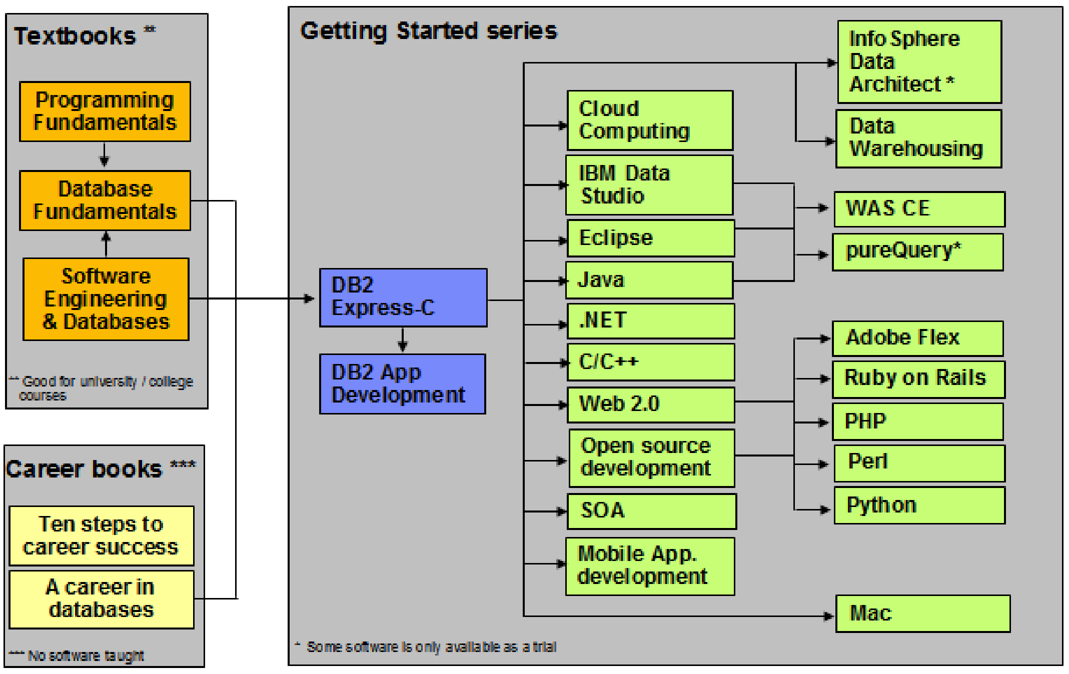
Italics are also used for variable names in the syntax of a command or statement. If the variable name has more than one word, it is joined with an underscore. For example:
`CREATE TABLE` ***`table_name`***

What's next?

We recommend you to review the following books in this book series for more details about related topics:

- *Getting started with DB2 Express-C*
- *Getting started with Ruby on Rails*
- *Getting started with PHP*
- *Getting started with Python*
- *Getting started with Perl*

The following figure shows all the different eBooks in the DB2 on Campus book series available for free at ibm.com/db2/books



The DB2 on Campus book series

About the authors

Rachna Kapur is the development manager for open source technologies at the IBM India software labs. She works with a team driving value propositions for IBM® Data Servers in the open source world. She brings 8 years of experience developing drivers in the health care and database domain.

Mario Briggs is the architect for the open source offerings for the IBM Data Servers, which includes Ruby/Rails, Python/Django/SqlAlchemy, Hibernate, Spring, iBatis & PHP. Mario has almost 11 years of experience in software development with many years in the area of data access, relational engines and application-database performance.

Tapas Saha is a graduate in electronics and communication engineering from Techno India, Kolkata, India. He joined IBM in 2007, as an open source software developer. Presently, he is working as a functional verification tester for DB2® for Linux®, UNIX®, and Windows®.

Ulisses Araújo Costa is a software engineer finishing his MSc in Formal Methods and Intelligent Systems. He loves functional programming and open source methodologies. He is a pro-active individual who loves to be connected to academic initiatives, like the open source support center at the University of Minho and the DB2 student ambassador group at the same university.

Pedro Carvalho is a Software Engineering student at the University of Minho in Portugal. Aside from school, he has developed an interest in systems administration and team leading. He has been strongly engaged in local student organizations, mainly in the creation of an open source software group. He is currently working in a research scholarship in the area of language engineering and several Web development projects.

Raul F. Chong is the DB2 on Campus program manager based at the IBM Toronto Laboratory, and a DB2 technical evangelist. His main responsibility is to grow the DB2 community around the world. Raul joined IBM in 1997 and has held numerous positions in the company. As a DB2 consultant, Raul helped IBM business partners with migrations from other relational database management systems to DB2, as well as with database performance and application design issues. As a DB2 technical support specialist, Raul has helped resolve DB2 problems on the OS/390®, z/OS®, Linux®, UNIX® and Windows platforms. Raul has taught many DB2 workshops, has published numerous articles, and has contributed to the DB2 Certification exam tutorials. Raul has summarized many of his DB2 experiences through the years in his book *Understanding DB2 - Learning Visually with Examples 2nd Edition (ISBN-10: 0131580183)* for which he is the lead author. He has also co-authored the book *DB2 SQL PL Essential Guide for DB2 UDB on Linux, UNIX, Windows, i5/OS, and z/OS (ISBN* 0131477005), and is the project lead and co-author of many of the books in the DB2 on Campus book series.

Peter Kohlmann manages the DB2 for Linux, UNIX and Windows Product Planning Office. His team is responsible for requirements management, business analytics, tactical planning, technology demonstration and customer relationship management. He is also the original developer and the design lead for the Technology Explorer for IBM® DB2® open source project. Peter has been in database technology development since 1989. This includes writing about application development, database administration and performance tuning. He has worked as team lead, architect and development manager in user interface development for DB2. He managed the business development process for the Information Management division, and was the project manager for the first large scale TPCC performance benchmarks for DB2. Prior to his current role he was the DB2 Express development lead.

Contributors

The following people edited, reviewed, provided content, and contributed significantly to this book.

Contributor	Company/University	Position/Occupation	Contribution
Misato Sakamoto	IBM Toronto Lab	Technology Explorer for DB2 - Developer	Overall review and input for Chapter 8
Leon Katsnelson	IBM Toronto Lab	Program Director, IBM Data Servers	Technical review

Acknowledgements

We greatly thank the following individuals for their assistance in developing materials referenced in this book:

Martin Streicher for his article "Open source licensing" [1] published by IBM developerWorks®

Natasha Tolub for designing the cover of this book.

Susan Visser for assistance with publishing this book.

1

Chapter 1 – Introduction to open source development

Open source software development is a methodology for creating software products, from the design and development to distribution. Under this methodology the author offers access to the source code. This chapter gets you started into this fascinating software development world. It teaches you how it all started, and what the direction will be for the coming years.

In this chapter you will learn about:

- A brief history about open source software development
- The evolution of the open source movement
- Open source versus free software
- Advantages and disadvantages of open source
- Trends and perspectives
- Career prospective in the open source world

1.1 A brief history about open source development

The story of open source development started long before Richard Stallman created the Free Software movement. In the 50's and 60's almost all the software that existed was mostly produced by research institutes. Software was not seen as a product. In those times, computer companies were in the hardware business; and software was made freely available to encourage hardware sales. The source code was distributed with the software because users often had to change the code to fix bugs or add new features to support hardware issues. During this time software was developed and distributed by communities of user groups and no effort was needed to make it freely available.

Things started to change in the early 1970's when operating systems and compilers began to grow very fast, with the emergence of micro-processors. For almost a decade until the early 1980's computer vendors and software companies began to routinely charge for software licenses, and sell software as a product imposing legal restrictions on new software developments through copyrights, trademarks, and leasing contracts. At that time two different groups, both in United States, were establishing the roots of the current open source software (OSS) philosophy.

In the East coast, a programmer at the MIT Artificial Intelligence Lab launched the GNU Project and the Free Software Foundation, his name was Richard Stallman. On the West coast, the Computer Science Research Group (CSRG) of the University of California at Berkeley were improving the UNIX system, and started to build lots of applications which quickly became known as “BSD UNIX”. With the advent of Usenet, an Internet user group, programming communities started to share their software and contribute to each others' work. This is when the real development of the open source movement began.

1.2 The evolution of the open source movement

We start recording the evolution of open source development from the creation in 1986 of the Free Software Foundation by Richard Stallman (who likes to use his initials RMS). After this foundation was established, several major open source projects were initiated as shown in *Figure 1.1*.

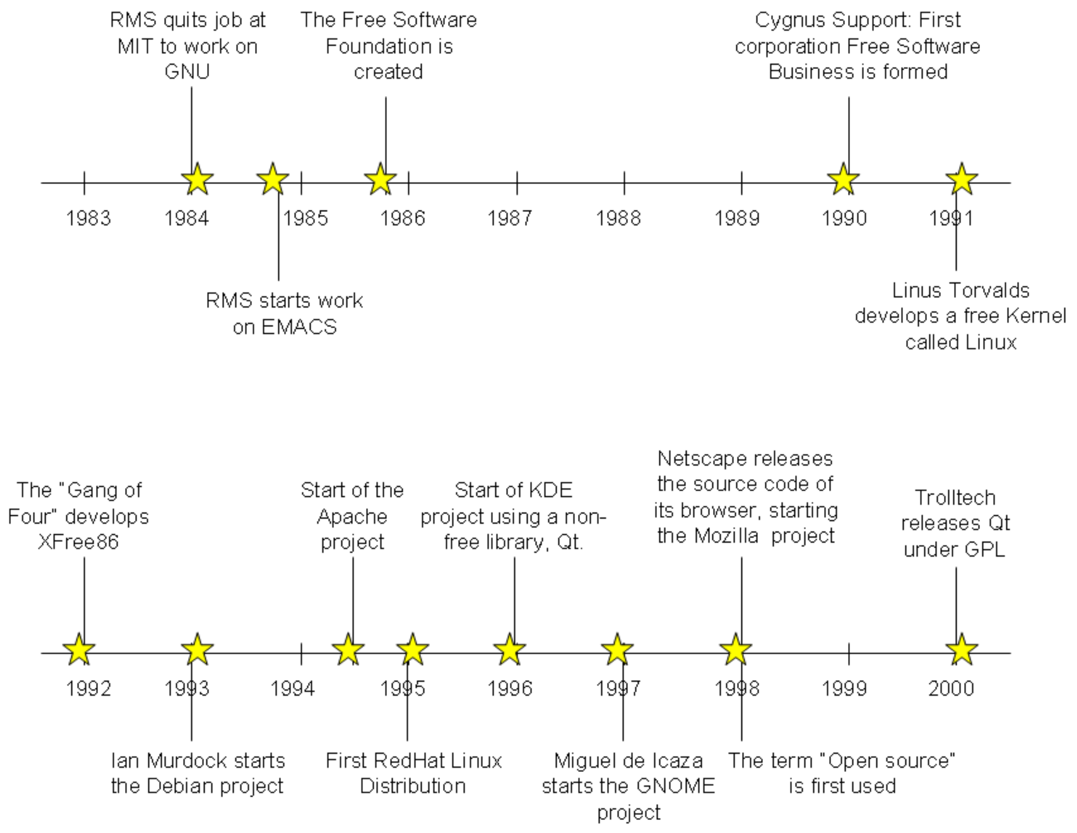

RMS quits job at MIT to work on GNU
The Free Software Foundation is created
Cygnus Support: First corporation Free Software Business is formed
1983
1984
1985
1986
1987
1988
1989
1990
1991
RMS starts work on EMACS
Linus Torvalds develops a free Kernel called Linux
The "Gang of Four" develops XFree86
Start of the Apache project
Start of KDE project using a non-free library, Qt.
Netscape releases the source code of its browser, starting the Mozilla project
Trolltech releases Qt under GPL
1992
1993
1994
1995
1996
1997
1998
1999
2000
Ian Murdock starts the Debian project
First RedHat Linux Distribution
Miguel de Icaza starts the GNOME project
The term "Open source" is first used

Figure 1.2 - Evolution of Open Source development

In 1984, Stallman started the GNU project with the ultimate goal of building a free operating system. Within this project, he wrote the EMACS editor, which is considered by many as the most powerful text editor.

In 1985 Stallman created the Free Software Foundation (FSF), a nonprofit organization dedicated to the elimination of restrictions on copying, redistribution, understanding and modification of computer programs. The organization's goal was to promote the development and use of open source software in all areas of computing, particularly in helping to develop the GNU operating system and its tools. Until 1990 the foundation was dedicated to writing more software. Today, it coordinates many independent open source projects, where the FSF's focus has shifted to the structural and legal aspects of the open source community. As a legal tool, the GNU General Public License (GPL) was designed not only to ensure that the software produced by GNU would remain free, but to promote the production of more and more open source software.

In 1987 Stallman created an open source compiler, the GNU C Compiler (GCC) with the idea to encourage more open source code contributions. Nowadays the GCC compiler is often chosen as the favorite to develop software that needs to be implemented in various types of hardware and supports C, C++, FORTRAN, Ada, Java™, Objective-C and Pascal.

For many years, employees of Cygnus Support, a small company founded by Michael Tiemann, David Vinayak Wallace, and John Gilmore were the maintainers of several key GNU software products, including the GNU Debugger (gdb) and GNU Binutils. This company was also one of the major contributors to the GCC project.

In 1991 a student of the Department of Computer Science at the University of Helsinki in Finland, Linus Torvalds, created with the help of several volunteer programmers through Usenet the Linux® kernel. Some time after, the kernel grew with the assistance of developers around the world as open source software. A few years later a full open source operating system was available and released as GNU/Linux.

In 1993 the Debian GNU/Linux operating system was created by Ian Murdock in order to assemble all the GNU tools that existed at the time and the Linux kernel. The Debian Project grew slowly and gained notoriety when the program dpkg was released. This program is the basis of the Debian package management system and is used to install, remove, and provide information on the Debian software packages (.deb).

One of the most important events in open source development happened in 1994 when Robert McCool developed the Apache HTTP server. This Web server played a key role in the development of the World Wide Web, and it was the first open source alternative to the Netscape Web server. Today, more than 100 million Web sites use Apache as their Web server of choice, as shown in *Figure 1.2*.

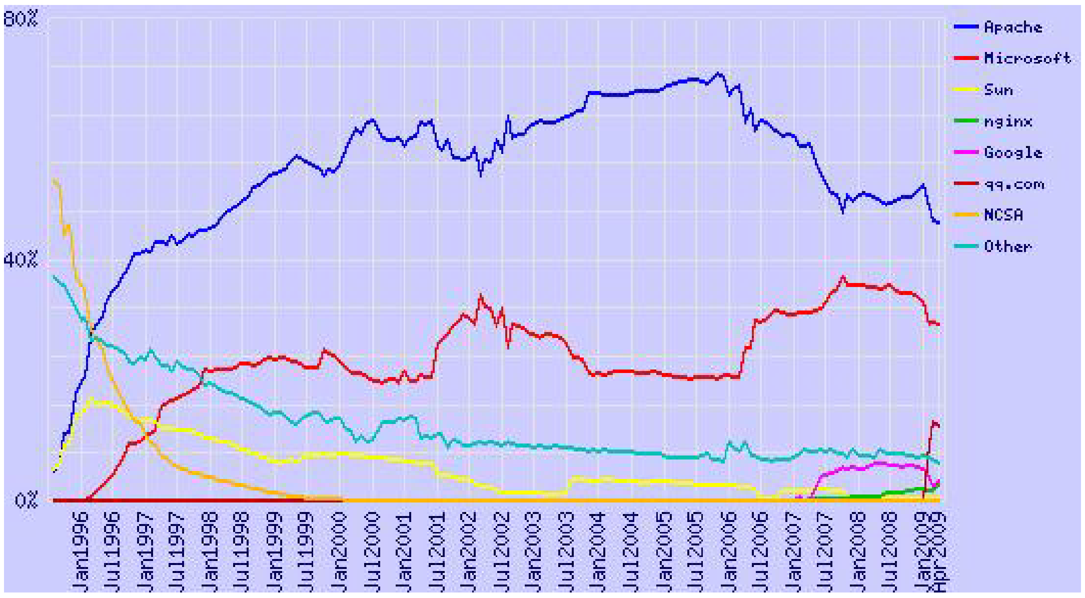


Figure 1.2 – Market share for top Web servers across all domains according to Netcraft (http://www.netcraft.com)

In 1995 Marc Ewing created his own Linux distribution called RedHat. RedHat is nowadays a major company that provides operating-system platforms along with middleware, applications, and management products, as well as support, training, and consulting services.

In 1996 the KDE and GNOME desktop environments were developed, providing basic desktop functionality for daily needs as well as development tools.

Two years later, in 1998, many software companies started to accept the open source movement when Netscape Communicator source code was made open source, and the Mozilla Foundation was established. It is in that year as well, that the term ***open source*** was first used at a conference in California. This term was created by Eric Raymond and Bruce Perens to promote the free software philosophy in the corporate world.

1.3 FLOSS - Free, *libre*, open source software

Until now the use of the term "open source" has many opponents. From one side Stallman and others with similar thoughts object to "open source" as they say it does not make users realize the freedom that the software in question gives to them. And when Stallman uses the term "free software", he is not referring to price as indicated by his quote, "*Free software is a matter of liberty, not price. To understand the concept, you should think of 'free' as in free speech, not as in 'free beer'*". On the other hand, using the term "free software" was commonly confused with the no-charge connotation, which obviously made business organizations uneasy.

In order to end this discussion other terms have been proposed, one of them is FLOSS – Free, *libre*, open source software, or Free (as in libre) open source software. There are three terms to explain this concept. As already mentioned, **free software** is a philosophical concept that aims to convey the idea of software that can be used, studied and modified without any restriction. It does not refer to price, but liberty. This software can be freely distributed. Because many people can confuse the concept of free software with freeware (closed-source software freely distributed by the author) the concept of Libre Software was created. **Libre** is a word used by several Romanic Languages such as Spanish, Portuguese and French. This word does not exist in English, but what it means is free as in "free speech", freedom as a right that a citizen has and not freedom associated with zero cost (Gratis). **Open source** on the other hand is a movement which emphasizes on source code of software being freely available. It also talks about certain criteria which must exist in the license of software to make it open source.

In this book when we used the term ***open source***, we mean FLOSS. In essence the freedom here refers to what an individual developer or user has with the code. It does not refer to a no-charge license. We discuss more details on the various connotations this phrase has in the open source world in the chapters to come.

Did you know?

The first use of the phrase “free open source software” on Usenet was in a posting on 18 March 1998, just a month after the term “open source” itself was introduced.

Did you know?

The IBM® DB2 Express-C database is free as in free beer. Check out these Web sites for more details:

http://freedb2.com/2009/08/06/db2-express-c-those-who-use-oracle-like-it-a-lot/

http://db2express.com

http://tldp.org/HOWTO/DB2-HOWTO/whyexpc9.html

1.4 Advantages and disadvantages of open source

Based on the fact that open source software is free and can be seen by everyone has great advantages. But the fact that it is being developed by a nonprofit community has some disadvantages. In this section we discuss the pros and cons of open source software maintenance and development.

1.4.1 Pros

Open source software has lower monetary costs as development, support and license costs are fairly minimal when compared to proprietary software. This does tempt many organizations to use open source software in their business model. In fact many companies do business with open source, IBM being one of them. For example, IBM offers a quality service with the Linux operating system. Another company, RedHat is selling the Linux operating system with support services.

As far as security and reliability goes, open source is a great model because, as many people analyze the source code, a safer and more secure code will be produced. Since open source software has so many people participating in its development, some programs are faster and scale better than the proprietary counterparts. Also depending on usage and need, open source software is also changed by experienced users thereby making code more stable.

Open source is also the answer to the incompatible formats in proprietary software, because it only uses open standards, that is, standards that are known or are accessible to all the people. One such example is OpenDocument Text (.odt), which is an open standard for word processor documents.

From a corporate perspective, companies that use open source do not have to worry about complicated licensing, and thus, do not suffer the risk of having illegal copies that infringe copyrights. Therefore, they don't need anti-piracy measures, such as CD keys, product activation and serial keys.

Open Source software is community driven and community serving; a large number of bright, and generous developers work openly and with the whole community. For example when an open source program crashes it provides useful information to find the source of the error or to report a possible bug.

Open source software is independent of companies and its main authors. If the company goes bankrupt or the authors fail to maintain the program, the code continues to belong to the community. Therefore, many people believe open source software can live by itself. As long as there are passionate contributors from the community, this is indeed the case.

1.4.2 Cons

Open source software has been focused to provide solutions to servers rather than to desktop computers. As a result, adoption in the desktop arena is much slower. For example, Linux desktops are still not used as much as Microsoft® Windows®. In addition, many software is not yet compatible with open source. When a user chooses a Linux desktop he has to remove several software that are not supported on Linux, or in some cases, there is no similar nor viable open source application. A good example is the gaming industry, which is still very focused on Windows.

Excluding companies that sell open source combined with technical support; proprietary software offers better service and support. The quality and availability of assistance in an open source project is proportional to the interest and use of the program by the community. An open source tool with few users can be poorly documented and have almost no means to help you understand it.

1.5 Open source trends and perspectives

The advantages of OSS outweigh its disadvantages, this is why companies are starting to pay close attention to open source. Increasingly, many companies are using open source software tools for development and test; but open source is quickly gaining market share as an alternative in production environments. A well known example is the LAMP stack – LAMP being Linux, Apache, MySQL® and PHP/Perl/Python. It is arguably the widest adopted software stack for the deployment of Web applications.

Many other open source projects that emerged since the 80's are still going strong, with many users and better product quality. This is the case of the Eclipse IDE (Integrated Development Environment). Eclipse was developed by IBM in 2001, and donated to the open source community in 2004. Today, Eclipse is one of the widest IDEs in use worldwide.

There are more and more companies whose business model relies on open source. For example, some companies specialize in delivering an integrated and fully tested environment consisting of different open source software like LAMP. The fact that such companies exist is the greatest measure of the success of open source community developed software.

Chapter 2 discusses in more detail other business models used in the open source world, and more details about Eclipse too. Other chapters in this book will discuss other successful open source projects in more detail.

1.6 Career path

We don't want to undermine your intelligence by making you believe that the number of jobs in OSS is much higher than that for proprietary software. However; though smaller, it is a niche market.

Today it would be easier to get a C/C++ programmer job. In order to get developers who have programmed in newer technologies like Python, or Ruby, and who have understood the community, its trends, and its ways of working would certainly take a lot more time and effort. Just as we make correct technology choices for a software solution based on its architecture, organizations need to make choices to find the right fit for the right job. With the growing trends of involvement and usage of OSS in companies the demand for open source developers to take up this work is definitely on the rise. IBM itself has shown a lot of involvement in open source software like Eclipse, Linux, and so on.

In summary, OSS jobs are increasing. It's still a niche area, but its growth potential makes it worth exploring. Needless to say that being an expert in a niche area always calls for a premium salary for oneself.

1.7 Exercises

Review the following link to learn about the involvement of IBM in OSS development.

http://www.accessmylibrary.com/article-1G1-133183812/history-ibm-open-source.html

1.8 Summary

In this chapter you learned about the history of open source and how it evolved over the years. You also learned about the underlying difference between the terms "free software" and "open source software". This moved us to discussing some of the advantages and disadvantages of open source software and how working in the open source space could benefit your career in the long term.

1.9 Review questions

1. When was the term "Open source" first used and by who?
2. What does FLOSS stands for?
3. List the pros of FLOSS
4. List the cons of FLOSS
5. Which foundation was started when Netscape Communicator software was made open source?
6. Which of the following open-source projects appears in the 90s?
   A. EMACS
   B. XFree86
   C. KDE
   D. Eclipse IDE
   E. None of the above
7. Who was the founder of the Free Software Foundation?
   A. Richard Stallman
   B. Ian Murdock
   C. Miguel de Icaza
   D. Linus Torvalds
   E. None of the above
8. One of the two groups who established the roots of OSS were:
   A. The founders of the GNU project
   B. The creators of BSD UNIX
   C. The founders of the GNOME project
   D. A & B
   E. None of the above
9. In Stallman's definition of Free Software, "Free" means:
   A. Free as in "Free beer"
   B. Free as in "Free speech"

   C. Free as in "Free of charge"
   D. All of the above
   E. None of the above

10. In Stallman's definition, Free Software is:
    A. Freeware
    B. Libre
    C. Software that can be modified, studied and distributed with no restriction, but may not be free of charge
    D. B and C
    E. None of the above

2

Chapter 2 – Open source business models

You've probably heard the saying "There’s no such thing as a free lunch". Do you think this applies to open source? Are the users of open source software getting a free lunch? The answer to this question lies in the famous quote "*free as in free speech, not as in free beer*" discussed in *Chapter 1*. If the creators of open source software are giving their users a free lunch, then how do they themselves earn a living or even be profitable? In this chapter we try to understand the answers to this question.

In this chapter you will learn about:

- The different business models employed by companies (not communities) involved in open source.
- How the business models affect you, as a contributor to open source, if any.

You might be asking yourself "Why should this matter to me as developer?" Strictly speaking, at this point you can choose not to bother; however, from our experience, this knowledge will help you understand some of the nuances behind why communities and companies do what they do. You may use this knowledge to your advantage, and perhaps use it as a guide towards your success when you start your own open source project! Understanding the economics behind how open source operates can be interesting in itself!

2.1 Open source business models: The big picture

An open source business model is a model used by companies that are involved in the development of open source software to keep themselves financially viable and successful. In fact today these companies compete with traditional proprietary software companies for investor’s money on the stock markets. Traditional software companies get revenue by the sale of the software they create, that is, they earn money for each copy of the software sold. As illustrated in *Figure 2.1,* traditional software business models monetize software by either directly selling software products or by providing software development services.

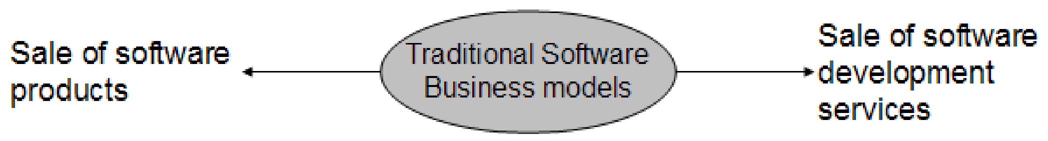

Sale of software products
Traditional Software Business models
Sale of software development services

Figure 2.1- Business models for Proprietary software

Did you know?

When IBM sold the first mainframes, the software that was bundled with them was made freely available in open source. Users were allowed to modify and enhance it.

How about open source software companies? Stepping back into the evolution of open source software, it is fair enough to say that the initial roots of open source software were sowed by either community projects which had mutual sharing as their main concern (over business ambitions) as in the case of the GNU project, or funded by government contracts as in the case of BSD UNIX. However as open source software usage progressed to the extent that many of them were viable alternatives to their commercial counterparts, commercial software companies became interested in finding ways by which they could promote, develop and monetize open source software. During this period many startup companies emerged, such as Red Hat. Red Hat focuses on the development and promotion of the Linux open source software, building their revenue and profits around it. These companies began to explore new economic models, different from traditional commercial software to succeed in the competitive software market.

This phase – companies involved in the development and promotion of open source software – has lasted for around a decade or more now. Today there are probably very few domains of product software from operating systems to Business Intelligence, in which there are no commercial companies promoting open source communities and their software. Studies have been carried out by groups to find out the various economic models employed by these companies. The top four models are illustrated in *Figure 2.2.*

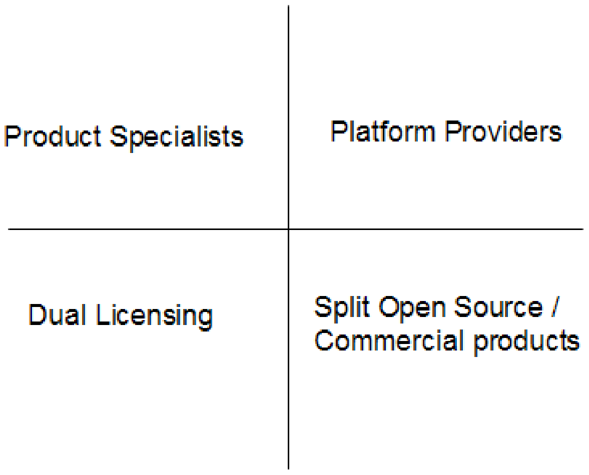


Figure 2.2- Business models for open source software

Companies that develop and promote open source software are sometimes referred to as ***Commercial Open Source Software companies (COSS)*** or ***Professional Open Source Software companies (POSS)***.

The following sections describe in detail each of the four business models shown in *Figure 2.2*.

2.2 Dual licensing

In this model, the open source software is licensed by the POSS company under both, an open source license (using GPL only, a license to discuss in detail in *Chapter 3*) as well as a commercial license. In this model the POSS company generates revenue when it sells the open source software under a commercial license.

Why would a consumer of open source software pay the POSS company to obtain the same software which is also available free of charge? This is needed when the consumer wants to link his own proprietary software to the open source software, but does not want this to cause its proprietary software to become open source as it would under the GPL license. According to the GPL license, when one accepts GPL licensed source code and links it with any other code (dynamic linking or static linking), the linked software also becomes open source. Thus the only way proprietary software vendors can link with GPL software without causing their own software to become GPL is by paying the POSS company. The POSS company then gives the same software to the proprietary software company under a license that excludes the need for the latter to make their proprietary software open source when they link to it. Figure 2.3 summarizes how dual licensing business model works.

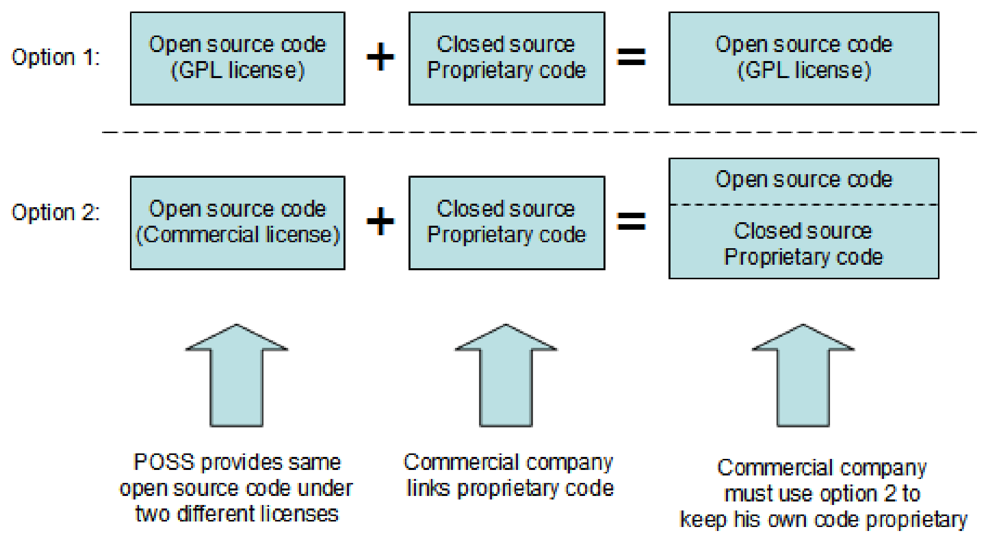

Option 1:
Open source code (GPL license)
+
Closed source Proprietary code
=
Open source code (GPL license)
Option 2:
Open source code (Commercial license)
+
Closed source Proprietary code
=
Open source code
Closed source Proprietary code
POSS provides same open source code under two different licenses
Commercial company links proprietary code
Commercial company must use option 2 to keep his own code proprietary

Figure 2.3 - Dual licensing business model

As a developer, you may want to consider this if you are planning to make code contributions to an open source project that has dual license. You will be giving the POSS company the right to make money directly out of your contribution by commercially licensing your code. Of course if the POSS company plans to accept your contributions they will ask you to sign a legal document which states that you have given them the rights to do so.

MySQL, JBoss and SugarCRM are examples of POSS companies that employ the dual licensing business model.

2.3 Split open source software / commercial products

In this model, there is a core portion of the software which is available as open source. This core provides the base functionality. Then there are other portions or extensions that are built on top of or extend the functionality provided by the core. The latter is licensed under commercial or proprietary licenses and sold like typical proprietary software. *Figure 2.4* illustrates how this model works.

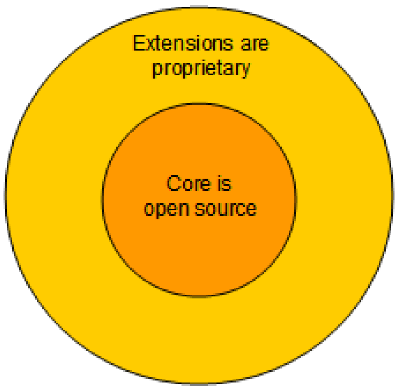


Figure 2.4 - Split open source software / commercial products business model

Many POSS companies choose the Apache or Mozilla open source license when they want to follow this business model, since these licenses allow this kind of intermixing where some parts can be open source and other parts proprietary.

This model is used by many commercial software companies that participate in open source including IBM. In fact, IBM is an acknowledged leader in this space. Does this mean IBM is a POSS company? A company is considered a POSS when most of its revenue comes from open source software sales. Since this is not the case for IBM it would not be considered a POSS company by that definition. Probably the best example of IBM's successful usage of this open source business model is Eclipse – the world's default IDE and tooling platform.

As discussed in *Chapter 1*, before IBM donated Eclipse to the open source community, a number of proprietary Java IDE's competed with each other; yet as a whole, Java IDE's had a very small portion of the IDE market share compared to Microsoft's Visual Studio. When IBM open sourced its VisualAge family of IDE products as Eclipse, it became the de-facto standard and was widely accepted by all. This led to a massive increase in the overall Java IDE market share compared to Visual Studio. Thus, IBM chose the right strategy with Eclipse, and is probably the biggest beneficiary financially - IBM Rational's development tools which are commercial software products have seen a great uptake since they are based on Eclipse.

If you are a developer planning to contribute to an open source project under this business model, you can be certain that your code will never be licensed under a commercial license, since it would be part of the core, which must be open source.

2.4 Product specialists

A POSS company that nurtured an open source software project either by creating it from the start or contributing and maintaining it, can generate revenue from it by providing training and consulting services to the customers of the open source software. This is illustrated in *Figure 2.5*.

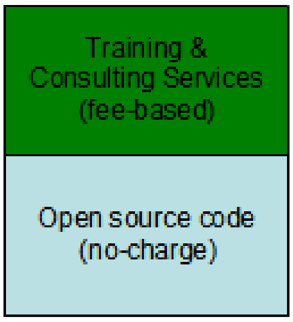

Training & Consulting Services (fee-based)
Open source code (no-charge)

Figure 2.5 - The product specialists' business model

If the open source software caters to a domain that is very complex, and it achieves good adoption, there can be significant revenue to be made from providing training and consulting services. For example, this may be the case with business intelligence software (as compared to wiki software). By the very nature of being the creators and maintainers of the software, it follows that they would be the experts, and hence the best trainers and consultants in the market. The POSS can achieve this position with minimal spending on marketing efforts.

2.5 Platform providers

In a not so distant past software systems were monolithic. In those days one probably bought the entire system from a single vendor who provided all the software and support needed. However these are the days of concepts like Service Oriented Architecture (SOA) where software systems are built from multiple components from different software vendors and integrated into one system. And what if those multiple components were all open source?

Open source software has permeated every domain of product software. Today many software systems in production run entire software stacks consisting of open source software. However, as you can imagine, customers building this type of systems have to face many challenging decisions: Which open source software receives community support? Which ones have a diminishing number of contributors and is on the way out?. How can you determine if a given open source software will work well with another one? How can these software be integrated taking into account different versions? How many resources should be allocated to test and verify the integrated system and then upgrade and patch it in the right manner so that it still works?.

Finding the right answer to these questions can be a nightmare. For example let's say that a patch of open source software X needs Java 1.5; however, open source software Y, which is also part of the system will work only with Java 1.4.

By now, you probably understand why a customer would be willing to pay a company to deliver a tested and verified system made up of different open source software. This is the reason the platform providers business model came into existence. *Figure 2.6* illustrates this idea.

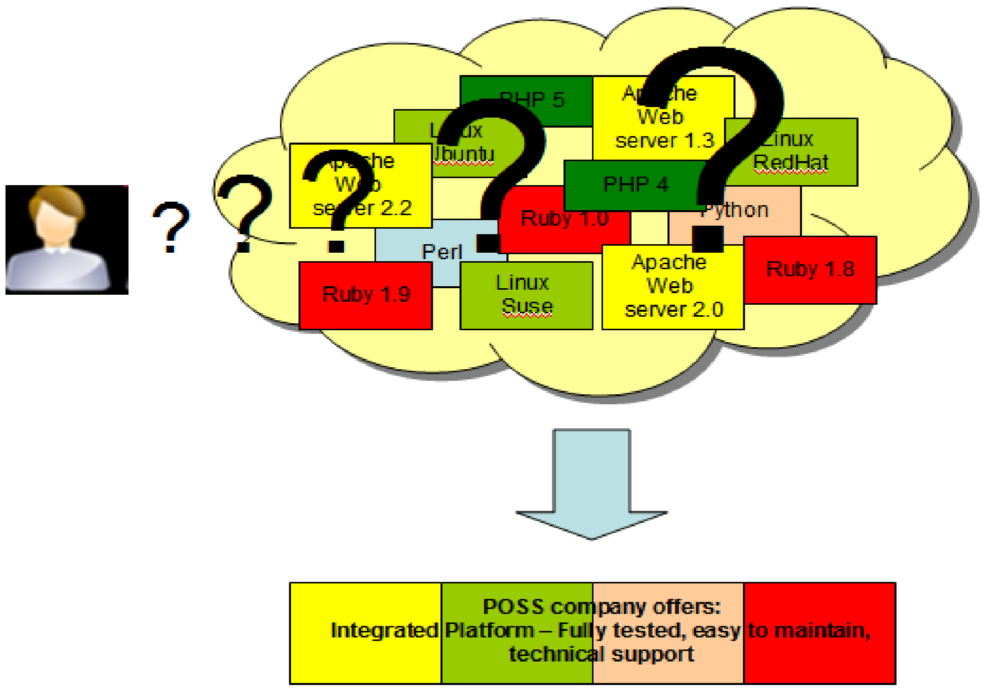


Figure 2.6 - The platform providers' business model

An example of a POSS company using this business model is Zend, "the PHP company". Zend provides a platform fully tested to develop PHP applications.

2.6 Business model relationship to license

It is interesting that when we analyze the above mentioned business models, the relationship these models have is vis-à-vis to open source licenses. This is illustrated in *Figure 2.7*. In the bottom quadrants, the business model is tied to the license of the open source software. Dual licensing applies only when the open source software is GPL-licensed, and Split Open Source / Commercial products when the open source software is Mozilla-based. However the business models in the top quadrant have no bearing on the license of the open source software itself.

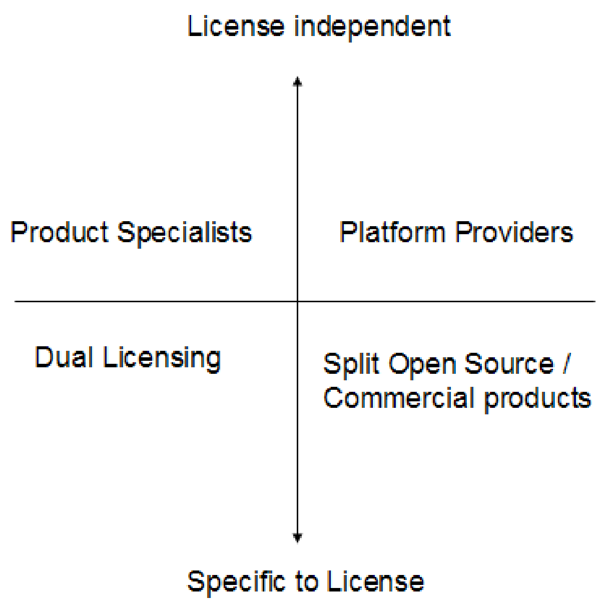


Figure 2.7 - Business model relationship to open source software license

2.7 Open source business model and proprietary software

Customers, startup companies and IT departments are finding it difficult to justify the licensing costs of proprietary software they use, and have been looking at open source software as a cost-reduction alternative. This has taken open source software adoption to new levels especially in segments like Web applications.

In order to remain competitive, several proprietary software vendors have evolved their strategies in this area. A widely used strategy is to offer no-charge versions of their fee-based products. The source code of these no-charge versions remains close. A typical example is with Database Management Systems (DBMS). Established DBMS vendors such as IBM, Oracle® and Microsoft compete today with open source DBMSs like MySQL and PostgreSQL by offering "Express" editions of their popular fee-based DBMS systems. The Express editions are free of charge, and by default have no customer support. This means that if you run into a problem such as a defect in the product, the vendor will not provide you with any support. However, vendors often optionally offer support for a fee. Thus proprietary software vendors are using the same approach as the open source business model – not charging for the software itself but generating revenue by providing support.

In addition to customer support revenue, proprietary vendors view this as an opportunity for adoption and upgrade to fee-based versions in the future. For example a startup company in its initial stages may not be in a position to spend lots of money to purchase software licenses; however, if the company grows, they would need more sophisticated product features which are generally not available in the free versions or on open source software. Therefore, an obvious choice for these companies would be to upgrade to a fee-based version of the proprietary software.

To an extent, Express editions have not seen significant adoption due to the customer's perception that these editions are nothing more than crippled software with upgrade paths to the fee-based versions. And though this may be true in some cases such as Oracle XE and Microsoft SQL Server® Express which place a 2GB hard limit on the amount of data that can be stored; other companies like IBM with its DB2 Express-C database product, do provide more flexibility.

Did you know?

IBM DB2 Express-C database server does not provide any restriction on the amount of data you can store. It can run on Linux, Windows, and the Mac OS, and can be installed on hardware of any size. Applications developed for DB2 Express-C will work with no modification on other editions of DB2, should you need to upgrade, because DB2 Express-C is built using the same code base as the other DB2 editions.

For more information visit ibm.com/db2/express

Did you know?

When Oracle bought Sun Microsystems in 2009, MySQL became an Oracle product since MySQL had been bought by Sun a few years earlier. At the time of writing, Oracle has not made it clear what are its plans for MySQL, but there are interesting comments here:

http://freedb2.com/2009/09/22/ellison-oracle-does-not-compete-with-mysql-mysql-disagrees/

In conclusion, using an Express version does have its merits when the product offers you low cost optional support, flexibility of use, and an easy upgrade path to more sophisticated editions in the future as your needs grow.

2.8 Summary

This chapter started with an explanation about the open source software ecosystem, and how it generates money to sustain itself. It then explained the different business models that exist in the world of open source, and how choosing a model can impact you as a contributor to open source software communities. Finally, we looked at "Express" versions of proprietary software as an alternative for companies to start with no licensing cost, the option for low cost customer support, and the possibility to easily upgrade to a version with more features.

2.9 Exercises

Review the following article which provides a brilliant analogy about a POSS company and bee-keeping.

http://wiki.pentaho.com/display/BEEKEEPER/The+Beekeeper

2.10 Review questions

1. Name a well-known database company that uses the dual licensing business model.
2. In which business model would the Eclipse project and its various plug-ins fall under?
3. What are Professional Open Source Software Companies?
4. Describe how DB2 Express-C is different from its competitors like Oracle XE or Microsoft SQL Server Express.
5. What is the value proposition from businesses using the Platform Providers' Business Model?
6. Which of the following is not one of the top business models used in open source:

   A. Dual licensing

   B. Platform Providers

   C. OS Providers

   D. All of the above

   E. None of the above
7. Which of the following business models require specific licenses?

   A. Split open source / Commercial products

   B. Platform providers

   C. OS Providers

   D. All of the above

   E. None of the above
8. Which of the following is a characteristic of DB2 Express-C?

   A. DB2 Express-C is a free database server by IBM

   B. DB2 Express-C has no limitation on the database size or the number of users

   C. DB2 Express-C can run on Linux, Windows and the Mac OS

   D. All of the above

   E. None of the above
9. What are the advantages of the express versions of some commercial products like DB2 Express-C?

   A. It's free

   B. It has optional fee-based support when your company grows

   C. It provides an easy upgrade path when your company grows

D. All of the above

E. None of the above

10. A company using the "Product Specialists" business model:

A. Does not spend a lot of money in marketing

B. Uses this model because it delivers complex OSS software in a large market

C. Relies on its expertise to deliver training given that it developed the software

D. All of the above

E. None of the above

3

Chapter 3 – Licensing

In earlier chapters you were introduced to the various business models employed by a company involved in open source, and several open source licenses were mentioned. But what makes a software license open source? This chapter provides an answer to this question.

In this chapter you will learn about:

- The tenets of intellectual property, copyright and the intents of an open source license.
- An explanation of what makes a license an open source license
- A brief description of some of the most commonly used open source licenses
- Things to keep in mind while choosing a license for a software

3.1 Intellectual property, copyright and licensing: The big picture

Intellectual Property (IP) are legally protected rights that one has over new ideas or creations. Common types of intellectual property include copyrights, trademarks, patents, industrial design rights and trade secrets.

Copyright gives the creator of the original work exclusive rights in terms of usage, distribution and customization of the work.

Some of the privileges copyright provides to the author of software include:

- The right to produce and sell copies of the work
- The right to create derivative works
- The right to sell, transfer or reassign any of the rights granted by copyright to others

The transfer of rights by the author partly or wholly at his own terms is what we refer to as ***licensing***. The term ***license*** means permission. The copyright holder, or ***licensor***, grants another person, known as the ***licensee***, specific permissions to use the work.

Figure 3.1 depicts this relationship between licensor, licensee and license. It also shows that a license is a subset of the privileges that the Copyright Act grants the licensor. The licensor permits the licensee to use these privileges as agreed between the two parties in the license.

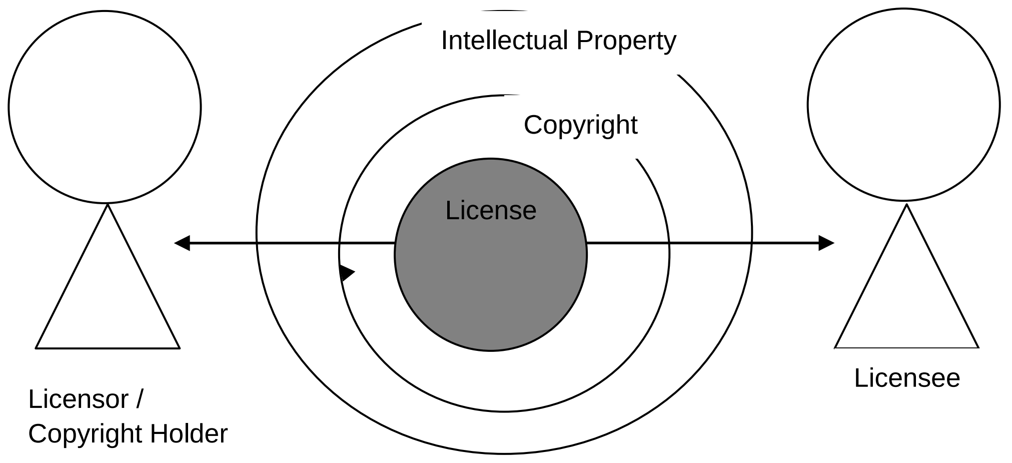


Figure 3.1: Relationship between licensor, licensee and license

3.2 Open source licensing

This section provides a brief history about open source licensing, its intent, and a brief description of commonly used licenses.

3.2.1 History of open source licensing

The open source software movement traces its history to the formation of the Free Software Foundation ("FSF") in 1985 by Richard Stallman. As discussed in *Chapter 1*, Free Software is more of an ideology that emphasizes the freedom users have with the source code and not with the price one pays for the software. In essence, free software is an attempt to guarantee certain rights for both, users and developers. These freedoms include:

- Freedom to execute the program for any reason
- Freedom to examine the source code, see how it works and change it to do what you would like it to do
- Freedom to redistribute the source and keep the money generated from it
- Freedom to modify and redistribute the modified source code

In order to guarantee these freedoms the GNU General Public License (GPL) was created. In short any software licensed under GPL must include the source code. Any modifications made to a GPL source code will also be licensed under GPL. This was to ensure that software once "opened" to the community could not be "closed" at a later time.

In 1998 a non-profit institution called Open Source Initiative (OSI) defined the term "open source software" to emphasize a break with the anti-business past associated with GNU to place a new emphasis in the community on the possibilities of extending the free software model to the commercial world. The OSI does not define a specific license as GPL but lays down the pre-requisites of the distribution terms of open source software. It thereby accepts various licenses whose distribution terms comply with the Open Source Definition (OSD). There are ten criteria mentioned at the OSI Web site (http://www.opensource.org/docs/osd). The main intentions for the OSD are described in the article "*Open source licensing, Part 1: The intent*" [1] by Martin Streicher and are summarized in *Table 3.1* below.

	Intention	**Explanation**
1	Licensees are free to use open source software for any purpose whatsoever.	This basically means that the licensee need not justify the usage of the open source software.
2	Licensees are free to make copies of open source software and are free to distribute those copies without payment of royalties to a licensor.	This implies that the licensee can redistribute the software at a cost or free of charge.
3	Licensees are free to create derivative works of open source software and are free to distribute those works without payment of royalties to a licensor.	This allows the licensee to modify the open source software and then redistribute it with or without charge. The licensee is in no way liable to pay any amount to the licensor for this neither can the licensor pose any restrictions on the derivative works. A pre-requisite for intention 3 to occur is the pre-existence of intention 4.
4	Licensees are free to access and use the source code of open source software.	This means the source code is freely available
5	Licensees are free to combine open source and other software.	This gives the licensee the ability to mix open source software with other software.

Table 3.1 - Intentions of open source software

We review in the next section how the GPL and MIT licenses meet the above intents of open source software.

3.2.2 Commonly used open source licenses

Though there are over 50 OSI approved licenses most of the licenses fall under two categories:

- **Academic licenses,** such as the Berkeley Software Distribution (BSD) license, allow software to be used for any purpose. Software obtained via an academic license can be freely changed, sold, redistributed, sublicensed, and combined with other software.
- **Reciprocal licenses** like the GNU General Public License (GPL), also allow software to be used for any purpose, however it enforces that the changed or modified software must also be licensed under the exact terms as the original license.

A GPL licensed code does not allow proprietary software to link to it. It also does not permit redistribution with software having a GPL non-compatible license. Also redistribution of the derivative works need to be with GPL. On the other hand MIT licensed software allows all of it. It permits proprietary code to link to it, redistribution with non-MIT license software and redistribution of derivative works with non-MIT license. Interestingly, they both are open source software licenses as they follow the open source definition specified by the OSI. *Table 3.2* compares the GPL versus the MIT license.

	GPL license	**MIT license**
Allow proprietary code to link to open source code	No	Yes
Allow redistribution of software with other code that has another license	No	Yes
Allow redistribution of derivative work	Yes. Derivative work becomes open source with GPL license	Yes

Table 3.2 - Comparing the GPL vs. the MIT reciprocal licenses

Let's take an even closer look at the GPL and MIT licenses by reviewing excerpts of each license as shown in *Listing 3.1* and *3.2* respectively. We will verify each of the licenses satisfy the five intentions explained earlier in *Table 3.1*

You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work -- provided that you -- cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works.

But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Listing 3.1 - An excerpt of the GNU license

The sentence *"You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work"* indicates that licensees are free to access, and use open source software for any purpose and that they are free to create and distribute derivative works (Intention 1, 3, 4 and 5). In the same sentence, the clause *"copy and distribute such modifications or work"* further indicates that licensees are free to make copies of the open source software (Intention 2).

Copyright (c) year, copyright holders
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

< text that follows has been removed for the sake of brevity >

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

Listing 3.2 - An excerpt of the MIT license

The phrase *"permission is hereby granted, free of charge"* clearly indicates that MIT permits licensees to access and use open source software for any purpose whatsoever (Intention 1). Further the phrase *"rights to use, copy, modify, merge, publish, distribute, sublicense and/or sell copies"* combined with *"free of charge"* indicates that licensees are free to make and distribute copies, access the source code, create derivative works and combine open source and other software without payment of royalties (Intention 2, 3, 4 and 5).

In conclusion, both GPL and MIT are valid open source licenses as they satisfy the same intentions though through different means, and are thereby best suited in different situations.

Did you know?

The IBM Public License (IPL) was IBM's first open source license. The Common Public License (CPL) is essentially the next version of the IPL. In 2009 CPL was superseded by the Eclipse Public License (EPL).

3.3 Choosing the right license

By now you must be feeling more of a lawyer and less of a developer. Yet, all of this information is very relevant to you: It can help you avoid a lawsuit. An interesting licensing violation is the case between the Free Software Foundation (FSF) and Cisco. In 2004 a German court ordered Cisco to stop selling its wireless routers because it was in violation of the terms and conditions of the GPL license - Cisco's products were using GPL-licensed software in their code, but not providing free availability of their source code.

Another case of license violation happened early in 2009 when Microsoft was found in violation of the GPL license on the Hyper-V code it released to the open source community. More information can be found at this link:

http://www.theregister.co.uk/2009/07/23/microsoft_hyperv_gpl_violation/.

So what is the right open source license to use for a given software? There is no magic formula but some of the things to consider are listed below:

- Is the work in question a derived work? Or would the work be redistributed with other open source software? If so, what are the license terms you have agreed as a licensee?
- Do you want the software to be revenue generating? Review the different open source business models as described earlier in *Chapter 2*.
- Do you want access to use licensees' contributions? How do you add this clause to your license?
- Review the licenses in the OSI website - http://www.opensource.org/licenses to see if any of the existing and approved licenses of OSI would be good for you to use.

This and several other factors are things to consider while choosing an appropriate license for the software in question. Depending on the scope of the project you could get into legal discussions to help you chose an appropriate license.

3.4 Exercises

The OSI categorizes various approved licenses. Review these categories and the licenses' terms and conditions at http://www.opensource.org/licenses/category

3.5 Summary

In this chapter you learned about the concept of Intellectual Property and how it related to a license. You also learned about the relationship between a license, licensor and licensee. Next, the chapter explained what makes a license an open source license, and provided a description and comparison between the BSD, GPL and MIT licenses. Finally the chapter provided questions that would help you choose the most appropriate open source license for your needs.

3.6 Review questions

1. What is required for OSI license approval?
2. Why is it important for developers to know about OSS licenses?

3. Name 3 popular OSS licenses
4. How come the MIT and GPL licenses are both OSS licenses even though they have contradictory items?
5. Name the two main categories of OSS licenses
6. Is the Eclipse Public License approved by OSI?
    A. Yes
    B. No
7. If I modify a program licensed under GPL and distribute the object code of the modified program for free, must I make the source code available?
    A. Yes
    B. No
8. Under MIT, can you link proprietary code to open source code?
    A. Yes
    B. No
9. Intellectual property and copyright are synonyms. True or false?
    A. True
    B. False
10. Which of the following is not an OSS license?
    A. GPL
    B. BSD
    C. MIT
    D. EPL
    E. None of the above

4

Chapter 4 – Community driven development

Open source software is often developed by a group of people who normally have no governing authority or organizational support. What are the strategies and methods they follow? Which tools do they use at the various phases of the software development life cycle? Are these tools available for free? This chapter provides the answers to all of the above questions.

In this chapter you will learn about:

- Various tools used in an open source software development project.
- Processes of designing and writing the programs.
- Managing different versions of the source code and creating new builds.
- Methods of testing and issues tracking.
- How open source software is released.

4.1 Community driven development: The big picture

A ***community*** is a diverse group of people sharing their thoughts, ideas, work, and experiences, at a common place. The term "community" as used in this chapter is not an exception to this definition. In the software industry, ***community driven development*** broadly means an initiative under which a group of technologists work together with a common vision of producing open source software.

Most of the OSS development communities who are successful today, organize themselves in a similar way as a professional organization or proprietary software company. This is why in a well structured OSS community; you can find each member has a job role within the team. Groups of people performing similar jobs, form sub teams within the community. *Figure 4.1* gives you an overview of such a community, where multiple sub teams collaborate to develop open source software.

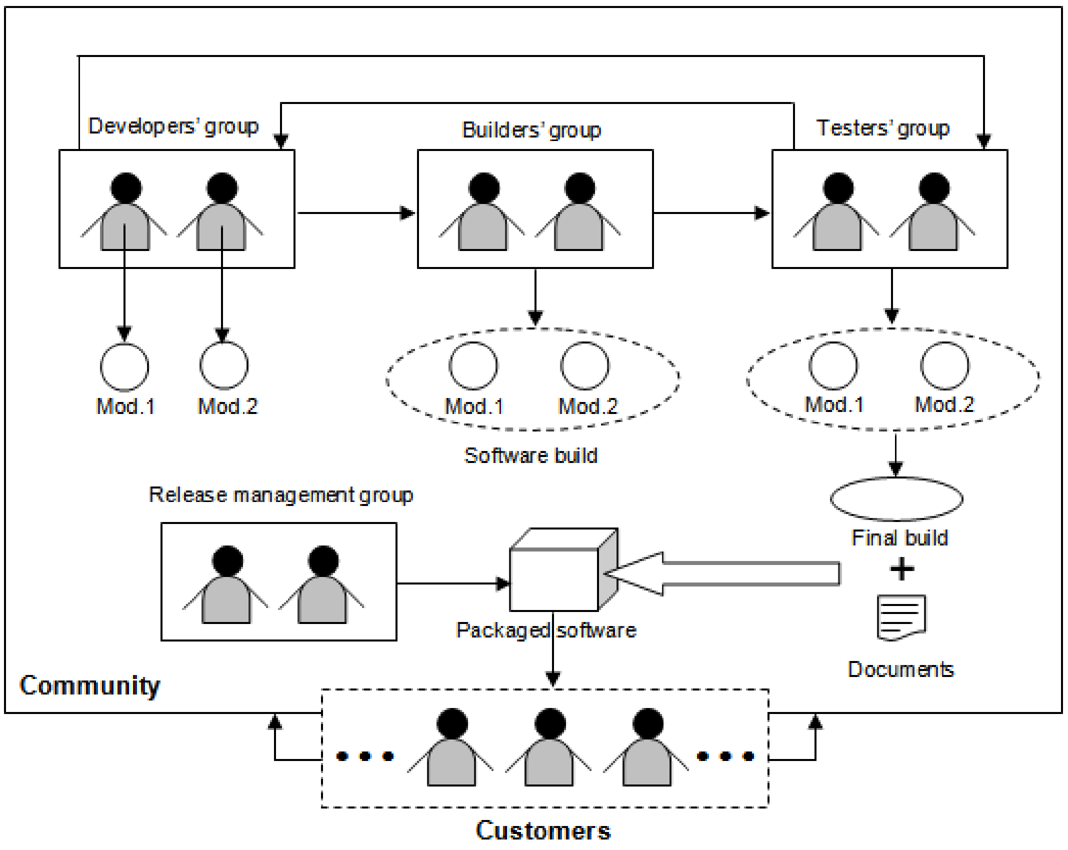


Figure 4.1 – Community driven software development

As shown in the figure, the ***developers' group*** is responsible for writing code for different independent modules of the software. The ***builders' group*** takes these modules from the developers and put them together to build a new version of the software. These software builds are internally tested by the ***testers' group***. In case a bug or failure is found, they report back to the appropriate developer who debugs the code and includes a proper fix. This is an iterative process that repeats itself until the product satisfies all its requirements and becomes error free to the maximum possible extent. Once the objectives of the project are met, the ***release management group*** packs together the final version (final build) of the software with all the necessary documents, and then hands it over to the customers.

OSS development communities, especially those which are self driven, offer their team members the flexibility of shuffling their job roles. This means that a developer may perform the role of a tester, a tester may work as a build team member, and so on. Most importantly, these communities always keep the door open for the users (customers), so that they can also contribute effectively into the project and become part of the community.

Software development tools are a collection of programs that help with specific tasks during the software development life cycle. They are normally used for project management, and to perform repetitive, and time consuming tasks. There exist both, proprietary and open source development tools. For an open source project, it should be obvious that the latter one is the preferred choice. There are also a few instances, where communities use their self developed tools.

The next sections describe the various processes and tools used by the OSS development communities in detail.

4.1.1 Developers' group: Software design and development

You can think of ***software design*** as the blueprint that defines the architecture and behavior of a product, and ***coding*** as the process of implementing the design.

A community driven software development typically starts as a project to address problems generic across the society. These problems are technically termed ***pain points***, and include a list of requirements that the software is expected to fulfill. The designers' group, which in many cases is also part of the developers' group, takes these requirements as input and writes algorithms which are cost effective and optimal in terms of performance.

Software designers should be the most experienced and knowledgeable members of the community as their job requires thorough understanding of the entire system. They also must be aware of every alternative to overcome all the probable barriers of implementing the design. Following are the two most important points that a designer must analyze closely:

- **Hardware platform:** The designer must know the capabilities of the hardware, with which the software is going to interact. Resources like number and speed of the processors, I/O device response time, available system cache, primary memory and secondary storage space etc. are the most important concerns.
- **Operating system:** The operating system (OS) provides the environment for running applications. Process and thread handling mechanisms vary from one OS to another. Moreover, the algorithms of job scheduling, CPU time allocation, avoiding deadlock situations are also different. The designer must take these factors into account.

A perfect design should be self explanatory and easy to understand. All the algorithms and flowcharts should include detailed explanations and other necessary documentations including time-space complexity and the expected behaviors of the system under best and worst scenarios.

Unified Modeling Language (UML) is a widely accepted standard used for software design. It allows a designer to describe the structure, behavior, and interactions among different sub units of the system with the help of various simple diagrams.

Following design, the software development life cycle enters the coding phase. At this stage, the design is first sub divided into multiple smaller units. Different development groups code these sub units separately as individual modules.

Coding is the process of translating the steps of an algorithm, into multiple computer instructions, with the help of a programming language. Choosing the proper language for writing the source code is critical. A community should always encourage using languages which have gained industry wide trust and with which the development team is well versed. Some communities prefer using interpreted languages, since they consume less time to create the executable modules as compared to the compiled languages.

Automatic code generators are often helpful in developing big and complex code snippets. *iCodeGenerator* is an open source code generator offered by SourceForge (http://sourceforge.net/projects/codegenerator/).

Transition between the design and code generation phases in a community driven development typically tend to overlap with each other in a project. Developers sometimes start coding parts of a system, while other parts are yet to be designed. Although, this kind of approach of running two phases in parallel saves time; it may result in duplication of effort in the event of a change in design.

4.1.1.1 Version control

Software development is an incremental process where functionality is added or removed one by one. At any given time the code of different modules may undergo frequent modifications by different individuals or groups which create different versions of the same code. ***Version control*** (or ***revision control***) is the mechanism to manage all of these versions in an effective way.

There are two varieties of version control systems:

- **Centralized Version Control System (CVCS)** – This is perhaps the most widely used version control system to-date. It works in a centralized manner which means that developers and testers around the world have to connect to a server, where a central repository for the entire project is stored.

  Concurrent Versions System (CVS) is a very well known open source CVCS (http://ftp.gnu.org/non-gnu/cvs/). To get a private copy of a particular file, you first need to check it out from the repository. Once done with your changes, you may commit or save your work into the repository by checking the file in. CVS will automatically increment the version (or revision) number by 1 and record all the necessary information about the change in its log. If someone else had already committed some other changes to the same file, CVS will automatically merge your changes with those of the other person. This system also allows a user to track back the old versions of a file. By comparing two successive versions of a file, you can easily determine modifications introduced.

  One of the main drawbacks of CVCS is that developers may interfere with each other's working environment. For example, a small bug injected through a code change, becomes a big issue when it causes various other modules of the entire code base to fail.

- **Decentralized / Distributed Version Control System (DVCS)** – This type of version control allows a developer or tester to create their own code branch, that is, they can maintain versions of the code in a decentralized manner. A DVCS provides all the functionalities of CVCS, plus the flexibility of creating a local repository. Git, Bazaar, Monotone, Mercurial and Darcs are a few examples of open source DVCS software.

Did you know?

During the early days of Linux® kernel development, Linus Torvalds used CVS for version management. In 2002, Linus switched to BitKeeper; however, since BitKeeper became proprietary in 2005, Linus started writing a completely new DVCS, called git. The official web site of git (http://git-scm.com/) describes it as – "*... a free & open source, distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Every Git clone is a full-fledged repository with complete history and full revision tracking capabilities, not dependent on network access or a central server. Branching and merging are fast and easy to do.*"

4.1.2 Builders' group: Software building

Programs are normally written in smaller units that are compiled before they can be executed. The resulting object modules are then bound together in a process known as ***linking*** to form a complete software system. ***Software building*** refers to the sum of ***compiling*** and ***linking*** all the source code of a project.

Software building is a dynamic process: People constantly add, delete and modify the source code files under the project's central repository. To make these changes effective, the modified code needs to be recompiled and linked regularly. Development communities generally have a separate build team to carry out this task. It is important to mention that the build process recompiles only those source code files which were modified since the previous build was released. For the rest of the elements which remained unchanged, the older compiled modules are picked up. *Figure 4.3* illustrates this fact.

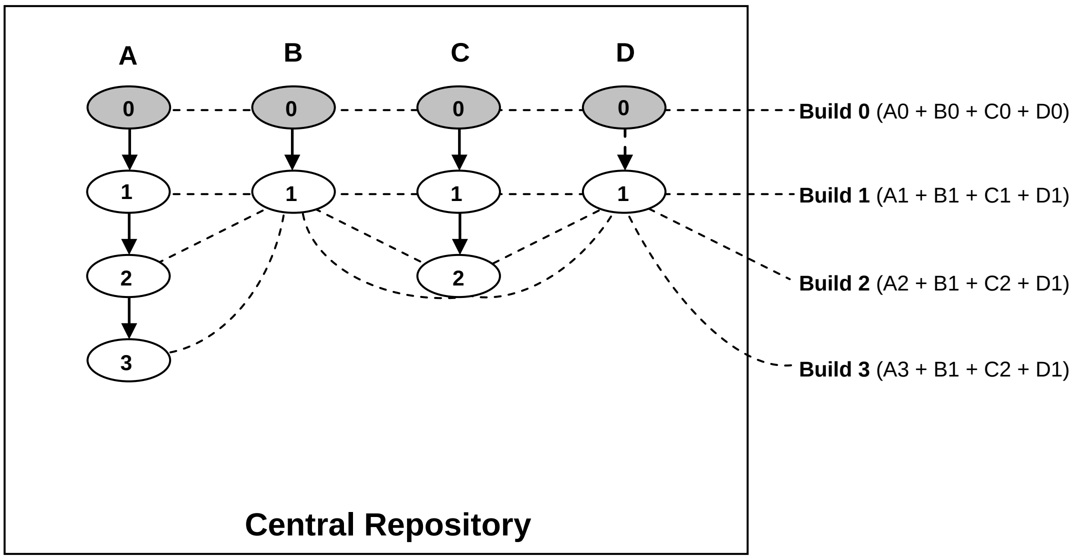

A
B
C
D
0
1
2
3
Build 0 (A0 + B0 + C0 + D0)
Build 1 (A1 + B1 + C1 + D1)
Build 2 (A2 + B1 + C2 + D1)
Build 3 (A3 + B1 + C2 + D1)
Central Repository

Figure 4.3 – Software builds

In the figure, the central repository contains four source code files A, B, C and D. The base version of these files is illustrated with the number zero, so A0 represents the base version of source code file A for example. When compiled and linked they create ‘Build 0’. On the next iteration, all the files were modified and their versions were upgraded to 1. Since all the files were modified, all are compiled and linked to create ‘Build 1’. On the net iteration, only source code file A and C are modified creating second versions of them (A2 and C2). As a result when creating ‘Build 2’, only A2 and C2 are compiled, and then linked with B1 and D1. Similarly, ‘Build 3’ is created based on the newly compiled A3 and precompiled B1, C2 and D1.

Berczuk and ***Appleton*** have divided software builds into the following three distinct categories in their book *“Software Configuration Management Patterns”* [2], and an additional paper with ***Konieczka***, *“Build Management for an Agile Team”* [3]:

- **Private build** – A private build is a new version of a developer’s personal branch in a DVCS. Naturally, this type of build is only visible to its owner and he can access it locally even when working offline.
- **Integration build** – An integration build is created from the compilation and linking of the main project branch and is available to all within the community for use.
- **Release build** – A release build is the final version of the software to be shipped. It is basically a snapshot of a stable integration build which users use as a product.

Frequent builds allow the source code to undergo less number of modifications. This reduces the effort needed for debugging and fixing the new errors. On the other hand, the build process involves a lot of repetitive tasks which can be automated by means of open source tools like *make*, *ant*, *maven*, and *scons*.

So far we have discussed about the building process in a project environment. However, since the code is open, at anytime users can tailor it at their own premises and get customized versions of the product. A later section discusses how you can rebuild and install open source software after changing its source code.

4.1.3 Testers' group: Software Testing

Software testing as defined by ***Glenford J. Myers*** in his book *“The Art of Software Testing”* [4], is *“the process of executing a program with the intent of finding errors"*.

Conventionally, there are two varieties of software testing – ***black box*** and ***white box***.

- **Black box testing:** In this type of testing, the actual output of the code is compared with the expected result. Testers are supposed to feed the system with a set of inputs, and note down the corresponding outputs. In case of any mismatch between the obtained and expected results they report it to the developer without analyzing the code.
- **White box testing:** In contrast to black box, in this type of testing, testers look into the source code of the product and carry out some investigation in the event of

failures. This requires a developer to put less effort and time for analyzing the error before fixing it.

With open source software, every person, no matter whether he is a member of the community or not, gets equal opportunity of performing white box testing as the source code is accessible to all.

Testing also requires planning and a proper infrastructure. At different stages of development, the code undergoes various levels of testing:

- ***Unit testing:*** This level is used to test individually each of the units that make up a software
- ***Integration testing:*** In this level, all the stable modules are integrated with each other to form the entire system, and tested as a whole.
- ***System testing:*** This testing ensures the software is operational and satisfying all its requirements.
- ***Alpha testing:*** In this stage, the software is given to the users internal to the community, to judge its performance against practical deployment.
- ***Beta testing***: After fixing the errors discovered during alpha testing, the community releases the software to the external users as a beta version, with a disclaimer that it may fail in unplanned use cases. The community expects external users to provide feedback about these unplanned scenarios.

Small scale open source software without any organizational governance may not undergo formal testing. However, this software tends to be quickly adopted because it provides a solution to a common problem. When a bug is encountered, the user will normally poke through the code to make things work again, or he can customize it to his own needs. This is the way in which many people become members of OSS communities. When the software becomes so useful and with enough followers, some users take the software to the next level by enhancing it, and formally releasing it to the community.

OSS development often spends less time and budget to test their product. Rather, they leverage the community to test and fix the code themselves; and to provide feedback for new functionality. In addition, this unique approach makes open source software very robust.

4.1.4 Release management group: Packaging

Users give preference to a product that is well packaged. In an open source software, the core item of a package is the source code. A standard package includes accessories like an installation guide, a user manual, and other information. Packages of software written in high level languages like C or C++ that require compilation of the source code, optionally may contain a pre-compiled executable version of the product. All of these elements are put together within a single directory and compressed into a single file.

The format of the compressed file depends on the operating system on which the software is meant to run. On Linux or UNIX®, packages are presented in formats like ***.tar***, ***.gz*** etc.

which are created using compression utilities like ***tar*, *gzip*, *bgzip*, *bzip2***. On Windows®, the distribution package is ***zip***. If the software has operating system specific versions, each of them is released in separate packages.

Once the package is prepared, it goes through a small test by a few developers to ensure the final version of the product can be decompressed, installed, and works correctly.

Then it is published in one or more internet Web sites for download. In some cases, if there is budget, CDs or DVDs can be ordered online at no charge.

4.1.5 Release management group: Releasing

Releasing software means making a completely new or upgraded version of a software available to the users. A new release introduces a product into the market and an upgraded version provides fixes for errors found in a previous released version and optionally adds new features.

To distinguish the versions from each other, each OSS community follows its own convention of using unique numbers for every release. Provided below is a template of such numbering scheme for the file names of the compressed package. The file extension is not shown, but it would be ***.tar, .gz, .zip*** and so on as discussed in the previous section:

```
Abcdefg x.y.z < Alpha | Beta >
```

In the above example, `Abcdefg` is the name of the software and `x`, `y`, and `z` represent the ***major, minor*** and ***micro*** components of the unique release number respectively. A release containing only small fixes for a previous version, is considered as a small or minor release and thus associated with the same release number as that of the previous one, incrementing its last digit (micro component) by 1. For example, if a product was released as `Abcdefg 2.0.5`, a small release would be represented as `Abcdefg 2.0.6`. A critical bug fix is implied by increasing the minor component and resetting the micro component which makes the new release to be identified as `Abcdefg 2.1.0`. If new functionalities are added, the version number would be upgraded to `Abcdefg 3.0.0`. When the software with limited functionality is made available to internal users for testing, this interim version is called Alpha as in `Abcdefg 3.0.0 Alpha.` An alpha version helps the development community collect error reports form the users. Depending on the count and sensitivity of the errors discovered, communities may also decide to release multiple alpha versions like `Abcdefg 3.0.0 Alpha 1`, `Abcdefg 3.0.0 Alpha 2`, and so on. When the alpha round is over, the feature complete product is given to third parties outside the community. This is called a `Beta` release. Following the example, the release would be called `Abcdefg 3.0.0 Beta`. With sufficient testing and debugging through alpha and beta cycles, the software finally becomes ready to be officially shipped. At this conclusive stage, the community announces ***"General Availability (GA)"*** of their product.

After a successful release, the developers' group work is divided in two: The first one is to maintain & support the already released version, and the second one is to further improve the product with new functionalities to be included in the next version. At this point, the central repository of the version control system is branched into two. The branch where maintenance works will occur is called the ***release branch***. Stable snapshots of this branch are released as minor versions of the product at the same major line (for example, `Abcdefg 3.1.0`, `Abcdefg 3.2.0` and so on). Fixes for high potential failures which cannot wait until next minor release are distributed to the customers via a sub branch of the current minor release line such as `Abcdefg 3.2.1`, `Abcdefg 3.2.2`, etc.

The branch where developers work on the new version (`Abcdefg 4.0.0` for this example) is called the ***mainline development branch.*** This branch progresses in parallel with the release branch.

A community should continue to provide fixes for previous versions until the majority of the existing users start using the latest version of the software.

4.2 Installation and issue tracking

After an OSS is released and is available to users, users have to install the product. In contrast to proprietary software, OSS may require users to compile the source code, and perform some configuration before the installation. This is described in this section. The section also describes how users can provide feedback to the community using issue tracking.

4.2.1 Installation

Before a user can install the software in his system, he needs to extract it by decompressing the package. Many users often attempt to install the software without reading the instructions. It is recommended to at least review the `README` or `INSTALL` files. Typical installation steps for open source software are:

- Understanding the system configuration
- Compiling the source code
- Executing the installation file.

For example, on a Linux or UNIX based system, the sequence of commands below accomplish these three steps:

```
# ./configure

# make

# make install
```

The first command is executed from the location where the software package is extracted. It invokes a tiny program called `configure` (generally included with the package) to detect the hardware and software specifications of the machine and create a `Makefile` containing the set of instructions to complete the installation process.

The second command, (**`make`**) compiles the source code of the software as dictated by the `Makefile`.

The third command (**`make install)`** reads the configuration specifications of the computer from the `Makefile` and installs the software into it. Although the first two commands can be executed by any normal user, the last one needs root access.

If the OSS package does not contain the `configure` script, users are provided with a `Makefile` they can edit with their computer's configuration and start the setup process from the second step.

Note that users can change the source code at any time and go through this process again to install the modified version of the product.

4.2.2 Issue tracking

Although the term "issue" may be interpreted as a bug or problem in the software development world, in reality it is more than that. An issue may refer to a bug, but also to a change request, where the change is related to a piece of existing code, design, feature or even software documentation.

OSS projects typically use free issue tracking tools like Abuky, Bugzilla, Buggit, BugRat, GNATS, JitterBug, and Mantis. An issue can be discovered by a developer, a tester or even a user. Once identified it is reported to the community by logging a record in the ***issue tracking database***, managed by the tool, with all the necessary details. Every single issue record is associated with a unique number for identification purpose. *Figure 4.2* shows different states of an issue throughout its entire life cycle.

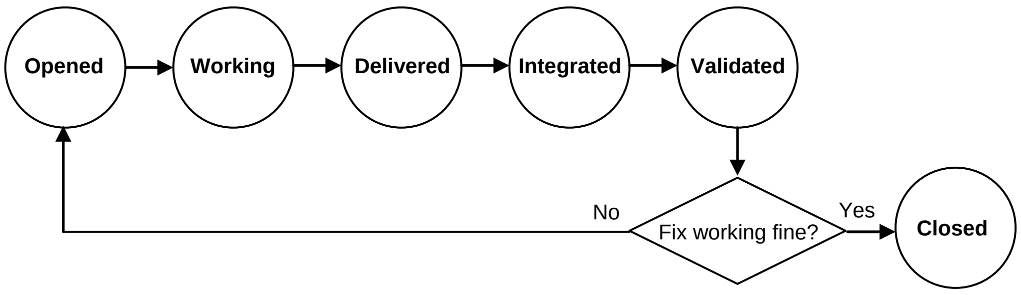


Figure 4.2 – Life cycle of an issue.

When a person ("the originator") finds a problem, he logs a defect in the issue tracking database, and the state of the issue is set to ***opened***. Thereafter, someone from the development team tries to reproduce the error and takes ownership of the issue if he validates it is indeed a bug. At that point the state of the issue is updated to ***working***. Once the developer (owner) fixes the problem, he commits the changes required into the central repository, and updates the state to ***delivered***. When these changes are included in the build of the project branch, the state changes to ***integrated***. If the build passes all the required tests, it is made available to the entire community and the issue state changes to

validated. The originator can now verify that the new build resolves the problem. If the error reoccurs, the state is switched back to ***opened*** and the whole cycle starts again; otherwise, the issue is ***closed***.

There are variations to the life cycle illustrated in *Figure 4.2*. It is common to find new ***opened*** issues being ***closed*** very soon due to these reasons:

- **Duplicate:** If someone opens an issue for an error which is already known to the development team, the issue is closed as a ***duplicate***.
- **Non-reproducible**: When an issue cannot be recreated under a similar environment, the issue is closed as ***non-reproducible***.

Such instances are very common as users not following an OSS project closely are also able to report failures. Moreover, there are instances where users open issues without understanding a feature properly. This can cause the issue tracking database of a popular OSS to grow considerably with invalid issues. To avoid this situation, it is the responsibility of the community's active members to follow up the issues as soon as they are opened by either accepting or declining them with proper justifications. On the hand, users should analyze a problem before opening an issue. They should at least perform some basic research by discussing the problem in blogs or discussion forums, or searching in the issue tracking database. Another option is to e-mail the development team directly, and only open an issue when they confirm it is a new problem.

4.3 Exercises

1. Visit Mozilla's bug tracking system at https://bugzilla.mozilla.org/ and see how an open source community tracks issues. Mozilla accepts bug reports and feature enhancement requests for all their products through this portal.
2. Know more about open source software testing from http://www.opensourcetesting.org/.
3. Learn about "Linux Software Packaging History and Concepts" from http://wiki.rpath.com/wiki/Appliance_Development:Linux_Software_Packaging_History_and_Concepts.
4. The Apache community uses Subversion (SVN) revision control system and allows everyone to access the source code over the internet. Know how you can get inside their code repository and work with the contents from http://www.apache.org/dev/version-control.html#source-code-repositories.

4.4 Summary

In this chapter, you learned how open source software is developed as a community initiative and how a project goes from an idea to a featured product. The chapter discussed all the stages of the software development life cycle, such as designing, coding, testing, building, packaging and releasing; with focus on the different groups who participate at each stage, and also on the open source software tools used.

4.5 Review questions

1. What does the term “community driven development” mean in the software industry?
2. State the key difference between CVCS and DVCS.
3. Name some open source issue tracking tools.
4. What are the items typically included within an OSS package?
5. What are the steps to install open source software in a computer?
6. Which of the following is a valid format of a Windows specific OSS package?

   A. .tar

   B. .zip

   C. .gz

   D. A and C

   E. All of the above.
7. Which of the following is used during the design process of an OSS?

   A. UML

   B. XML

   C. CVS

   D. Git

   E. None of the above
8. *myproduct 3.4.5* – in this scheme of version numbering, which is the “minor” component?

   A. myproduct

   B. 3

   C. 4

   D. 5

   E. None of the above
9. Which two of the following tools can be used to automate software build process?

   A. Make

   B. Apache

   C. Ant

   D. Darcs

   E. None of the above

10. Which of the following is <u>not</u> a state in the life cycle of an issue?

    A. Opened

    B. Fixed

    C. Closed

    D. Integrated

    E. Validated

5

Chapter 5 – Participating in open source development

On Chapter 4 you became acquainted with the software development process for OSS, and the relationship between the different groups in the community. This chapter is related to Chapter 4; however, the focus is more on the communication and social aspects when participating in an open source development project.

In this chapter you will learn about:

- How different communities operate, and how you can fit in
- Different channels for effective communication across the community
- How you should interact with others contributors and users

Note:

The writing style starting from this chapter is somewhat different than in the previous chapters. This is done on purpose. The writing style is used to also reflect the way many people in OSS communities communicate.

5.1 Participating in open source development: The big picture

Open source development is quickly gaining traction as one of the most effective development models. Just imagine the consequences of a having a handful of techies and savvy eyes going through your code, constantly reviewing it, correcting and adding to it. In little to no time your project would reach a level of quality you couldn't ever have imagined.

As discussed in *Chapter 4*, every major open source project has its own model of organization, a tight thought hierarchy where different people with different functions and different skills work in conjunction, most commonly across the Web. *Figure 5.1* illustrates a traditional organization of an open source community.

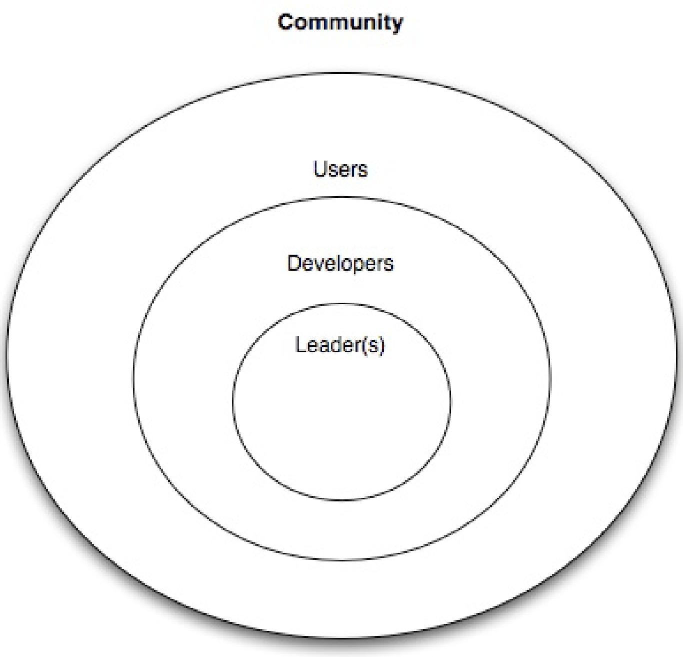


Figure 5.1: Traditional organization of an open source community

In *Chapter 4*, the focus was mainly on the groups responsible for the development of the software. In this chapter, we are taking a broader view. Notice how users take part in the organization as opposed to the closed software development method where users cannot take a more direct participation.

In order to contribute to an open source project a common misconception is that you have to be some kind of hacker; however, this is not the case. There are many ways one can contribute to an open source project.

Positions on a project vary according to project dimension; some projects behave much like a company. In those cases projects are divided into departments such as, development, public relationships, documentation, etc.; the structure reflects the project needs. What you should keep in mind is that there are many positions to occupy. In contrast, small projects are typically composed of a main developer with several co-developers. Though you are likely to be assigned your role when you first start contributing to a project, here is a description of the different roles normally available:

- **Project Leader / Main developer**: A leader performs functions that are not too different from a company's CEO: He is the one making tough decisions, and in big projects he is also in charge of coordinating many of the project subdivisions. Smaller projects often are governed by several developers, similar to a company's board of directors.

- **Programmer**: There aren't as many programmers as one would think; in many active projects the programmers are a small percentage of all contributors. The programmer is in charge of implementing features and fixing bugs.
- **Documentation Writer**: In the past open source software was difficult to install and only those who had the patience and skill could get this software working. This was perhaps one of the reasons why documentation became very important, as well as the fact that community members often rotate software development positions. It is the responsibility of the documentation writer to ensure that every person can use the software in hand.
- **Translator**: Even though English is the official Web language, it is true that not all of us are fluent in it, for that same reason it is common for a project to readily make its manuals available in as many languages as possible. Any user is more than welcome to translate any piece of the documentation to his own native language.
- **Designer**: Gone are the days when people in the Free and Open Source Software community didn't believe in image and its overall importance to a project. There are many kinds of designers. UI designers often called usability engineers, Web designers, and artwork designers. Projects like KDE or Gnome rely heavily on designers work.
- **Public relationship**: In some cases, such is the dimension of a project that huge amount of content is generated every day. Keeping the community up-to-date can be an overwhelming job. In order to keep a coherent image, a public relations team is assigned to key areas, passing the message along through the community communication channels.
- **Active User**: Active users of the software are the base of the pyramid. They use the software, provide feedback, and may customize the product to their needs.

The strength of the open source development model comes from the user base and the power given to it. A healthy project will empower users to submit feature requests, fill out bug reports, and will welcome user cooperation using properly administered communication channels. In a later section you will be introduced to some of the most used means of communication within a community, and you will learn effective ways of getting the word across.

5.2 Open source communities

Chances are you have found a certain software tool that satisfies your needs, but you don't have the money to pay for it. So you search through the Internet for an alternative free version, and after some time you find a free open source software that seems to have all that you need. Quickly you go through the features list and bummer!, an important feature that would be useful to you is missing. Most times someone with similar needs would have already gone through that same path and contributed with code that would have provided this specific feature; however, not this time; this time it's up to you!.

Contributing with code for the desired feature may take you weeks, but in the process you will learn tones of new and interesting information, gain a few ideas that would make that software snappier and meet people from all over the world who share the same passion you do; like the one of sitting alone at night with a stash of chips and a can of coke, cracking a problem like a mad man in pajamas, waiting for the moment the solution comes down that stairway from heaven, and rejoicing when it arrives by dancing with his cat.

Open source communities are full of people like this, well, maybe not all of them dance with their cats, but they love to be challenged and aren't afraid to get involved. For them opportunity often appears from necessity, which generates motivation which then mutates into fun and responsibility.

If you are reading this book, we assume you are interested in open source, and if you haven't already contributed in some way, maybe it's because you don't know where or how to start. Well, as many things in life, you have to find the need first; motivation is just not enough. By "need" we mean that you should search for a project you have a special interest in, for instance, start by looking for the ones you use on a day-to-day basis.

Take GNU/Linux for example. Many of us in the open source community use GNU/Linux daily, for it is a very robust, very powerful operating system, and with its many distributions it makes for a very good playground for people to enter the FLOSS community. Debian is one of those distributions, one of the oldest as a matter of fact, and it drives itself from a strict belief in free software as depicted in their "social contract", which guarantees you that in a bare installation, Debian contains only free software. Of the thousands of applications that make up Debian or almost any other distribution, many have their own community. Debian itself has a community.

It is then a matter of time that in your daily routine taming around your computer, you strike a bug in one of the thousands of software that make up Debian. Impaired and with no other alternative to continue whatever is you were doing, you look for a way to solve it. The bug hunting leads you to a little library with a very simple mission, intrigued you want to know more about that little library. Fortunately it is not hard in the open source world to find good documentation even to the simplest piece of software; good documentation guarantees project continuity and enables community feedback. Even small libraries like this one that is bugging you may have a community attached. It comes to no surprise, libraries are used by bigger applications, and the developers of these applications rely on these libraries for very specific functionalities; it's a mutual interest. If you are lucky, you may even find a bug tracking tool. For some of the smaller software projects that have no need for a bug tracking tool, the project leader normally goes with a simpler and centralized feedback platform, like a forum. It's very hard to be the first to find a bug in a community as large as Debian's, rest assure that if this example were to suit reality someone would have already fill in the bug for you. But in this reality you are the master, checking out every bug that bugs you. A little debugging with a proper software tool, a simple run through the source code and there it was. Five minutes later you have set up a patch, you go into the bug tracking tool and submit it, a few hours later one of the developers picks it up, merges it into the main code and redistributes it. Your good deed for the day is done.

Now don’t get the wrong idea, it is not all dull work and necessity that makes an open source contributor. Some of us, specially the younger ones, have a tendency to get bored quickly. There’s a little noise bubbling in our heads, like kids on Christmas Eve that just have a lot of curiosity. We go to where our current obsession drives us.

Finding an open source community that fits you out of pure interest should not be difficult, you see communities come in all size and shapes. They range from extremely large like the Linux kernel to the very small like your run for the mill script that downloads dubious content from the Internet. Some of the larger projects turn into non-profit organizations to provide a sustainable platform from where they can grow. This is why the Mozilla Foundation was born. You may know Mozilla as the organization who brings you Firefox, a really good Web browser that happens to be licensed as free and open source software.

These larger projects function very much like a typical company, structured into different departments, each with an individual area of operation.

Did you know?

In 1993 a company called Netscape released a version of a graphical Web browser named Mosaic. Mosaic was one of the first of its kind, revolutionary as a matter of fact, quickly becoming the world's most popular browser. In 1995 Microsoft launched Internet Explorer, becoming the first Mosaic opponent. Internet Explorer was bundled with Windows, which gave it quite an advantage. By the end of the 90’s Netscape was already in trouble. In order to save the company Netscape leadership created the Mozilla Foundation, and released their browser as open source. They expected that by releasing the code, a community would gather around it, helping the company regain the lost market. They were right!

Once you indentify a potential community with whom you’d like to get involved, you should take some time probing it. Try to figure how a new member’s work is contributed. Many projects will make available information on the subject but many others will not. If not in a separate section on their webpage, search forums for instructions on how to get your patches or bugs submitted. Go for frequently asked questions (FAQ) sections and wikis. Sometimes smaller projects include this information into text files that come with the source of the application, check it also! Ultimately if nothing else works search around the Web, maybe one of the community members has posted some information somewhere. A good example of a development guide is at http://ldn.linuxfoundation.org/how-participate-linux-community a Linux kernel developer's guide, which makes also a good read even if you don’t plan to take on kernel development.

Suffice is to say that open source has grown from being associated with a software project, to a more extended concept. Open source is now in effect a culture; it symbolizes free sharing of content. Communities gather around the most different concepts of content, written content, video, even hardware such as computer chips or cars are turned into open source.

Wikipedia is one of those communities, even though we must point out that technically it is not considered to be open source. This is because Wikipedia's license does not satisfy the open source principles; it is nonetheless a great example of peer contribution. You may even also come across some projects that have associated literature in the form of a book created gradually by contributors.

5.3 Effective communication

The internet is the foundation of open source communities since users and contributors are scattered across the globe. This also means that communication must be as effective and organized as possible. There must me established infrastructures, and these infrastructures must be well designed and should fully fulfill the community needs and modus operandi.

Though there are no standard communication channels, there is a historic means of communication used extensively across all open source projects: **email**. Email can be used as a great discussion mechanism through **mailing lists**. Mailing lists provide one of the most important community feedback channels, they are filled with developers taking on enthusiastic and technical discussions. There is also a more direct communication approach used extensively across the open source world, **Internet Relay Chat (IRC)** for instantaneous textual communication between two or more parties.

In the past BBSs (Bulletin Board Systems) were used to exchange messages with other users. In 1998 IRCs were introduced, and quickly became the de-facto communication tool for real-time communication throughout the 90's. IRC users have to install a client part of the software to their local computers. Today real-time communication falls in the hands of instant messaging services, but IRCs remain widely used. Due to its open nature it is a good way to meet other people. There are a number of IRC servers you can connect to, but the one used by most open source fellows are the freenode servers (freenode.com). A popular open source IRC is **mIRC** (http://www.mirc.com/).

Web sites are the first and foremost medium for promoting a given open source software and to mass communicate project details. You can host your own Web site with the help of open source Web servers like **Apache** (http://www.apache.org/). If Web hosting is not an area which you or your community has much expertise or interest in, you may consider using a site like **SourceForge**. SourceForge not only provides you with Web hosting, but also offers an integrated platform on which you can run your entire OSS project. For more information visit http://sourceforge.net/. In addition, the last chapter of this book includes a case study using SourceForge.

Did you know?

Part of the ranking process on sourceForge comes from downloads but much of the calculation is driven by community activity. If someone opens a defect, checks in code, makes a feature request, or posts to a forum page; all of that adds to the relative activity score and the project's overall ranking. Projects with a high ranking get noticed.

An example of a successful open source project hosted on SourceForge is the ***Technology Explorer for IBM DB2 (TE)***. TE has over 10,000 downloads with a sourceforge.org ranking consistently in the top 200 projects and in the top ten of over 13,000 database related projects. To find out more about the Technology Explorer for IBM DB2 refer to *Chapter 8*.

A ***Wiki*** is a collaborative Web site developed and maintained by a group of people for knowledge sharing purpose. In an OSS community, active members who are aware of the various technical details of the project, generally publish their own articles in the form of tutorials. http://wiki.rubyonrails.org/ is an example of such a wiki maintained by the Ruby on Rails user group. Wikis are generally open to all for contribution, with restriction only to the deletion of pages.

Feeling the pulse of the community is not restricted to their official communication channels. **Weblogs** (also known as **blogs**) have emerged has one of the most valid means of sharing thoughts around the Web. You are guaranteed to gather a lot of helpful information through blog surfing.

Planets are aggregation of blogs. Every major community has one; it's like a newsletter but much more effective. Developers write about their work and you get to know what's happening from an insider's point of view. It is also one great way members can gain visibility in the community. After all, being credited for great work is the only and best payback community members receive; and everyone likes to be appreciated for their work.

Still the best way by far for feeling your community is to meet the members in person. The most active and passionate members that happen to share the same geographic location organize themselves once in a while, to go out and get to know each other. These are called **user groups**. Look for user groups in your area and go have some fun!. Meetup.com is a Web site where you can easily arrange a local reunion with a specific theme. The Ruby community has done this. Check it out!, maybe there is one for yours already.

Some communities go further and organize worldwide conferences, where they can calmly and effectively discuss the future of the project as well as throw some technical presentations and boost the morale of everyone involved.

There are other types of meetings, however. FOSDEM, short for Free and Open Source Development European Meeting, is a once in a year event in Brussels. It gathers roughly five thousand visitors each year. Developers from all major open source projects come together to exchange knowledge which help promote their community and announce upcoming changes and innovations.

Last but not least, there are events called ***hackatons*** where developers come together and try to implement as many features or close as many bugs as they can. As you can imagine this can provide big pushes into a project's life cycle and as a side effect it improves collaborative working and respect among developers. Now, if you bring a bunch of developers together, give them a beer and a laptop, and you will certainly have a hackaton going as part of the original event.

5.3.1 Communication etiquette and guidelines

As passionate as people are in this kind of projects, it is not uncommon for opinions to diverge and matters to get out of hand. For that reason communities often follow guidelines for user behavior. These guidelines emphasize users' and contributors' communication conventions as to maintain social order and effectiveness.

These are some of the community guidelines:

- Search before you ask.

  This principle is quite simple, sometimes the information you are looking for is clearly stated somewhere within the project documentation or commonly available around the internet. It may be easier to ask on a mailing list or forum but this would be the equivalent to asking someone else to do the work for you.

- Do not engage in acts of verbal aggression.

  The Internet is a wonderful place, it enables people to communicate across continents with no connection whatsoever to their real identity, this triggers some primal human behaviors issues in some of us. Enabling these behaviors in yourself will not bring anything helpful to the conversation, it will however destroy any credibility you have left.

- Do not support acts of verbal aggression.

  If you are taking part on a discussion when all of a sudden a flame war erupts, please do not take part on it, respect other people’s opinions, be constructive and try to positively explain your view. Do remember a hierarchy exists, as in your professional life, you’ll often find people with more authority and lesser technical expertise.

- Know were to ask your questions.

  Usually a quick look around the project Web site is all you need to find your way to a forum or IRC channel where you can ask a specific question. Sometimes in bigger projects these platforms are carefully divided into different and appropriate sections where your question can be answered by someone with appropriate knowledge.

- Show results, you'll earn more respect showing work than talking about it.

  Often when you take part on a technical discussion and your point of view is not getting across, do not stress it, sometimes the best way to do it is by presenting the end result. Even if it turns out it is not the best solution, you will end up helping in achieving it, hopefully you'll learn something new as well, and you'll have proven yourself to the community.

- Be concise, explain your intentions first.

  Objectiveness is a big plus in any work area, besides saving you and others a lot of time; it simplifies the chore at hand. Across any written medium such as the ones we have seen earlier, it is absolutely essential that you make yourself and your intentions clear in any communication attempt. Start by asking yourself "is this information relevant?"

5.4 Exercises

1. Pick one or more OSS projects mentioned in this book so far, then choose your favorite search engine, and perform the following.

   A. Investigate the project purpose

   B. Find where it is heading, and where it came from

   C. Search for getting started documentation

   Note:

   If you would like to skip this step, Chapter 8 provides an example of an open source project where all the above information is provided.

2. Consider reading the essay "How to ask smart questions" by Eric Raymond on his FAQ page http://www.catb.org/esr/faqs/.

3. Choose the project you enjoyed the most from question one, keep in mind that it must have a forum or IRC channel. Now from what you have read in question two, you are more than able to engage in your community. Go to the community site and join their forum or IRC channel. Participate!

5.5 Summary

This chapter started discussing the motivations for contributing to an open source project. It also described the different roles a new member can take, which are not restricted to coding. The chapter then discussed a range of communication channels used by communities to exchange information; and finally it explained some guidelines and etiquette required when dealing with other members in the community, such as "Do not engage in acts of verbal aggression".

5.6 Review questions

1. How is a project ranked in sourceForge?
2. List some of the duties of a Project Leader.
3. Why is documentation important in an OSS project?

4. Why are there translators in an OSS community?
5. Name two communication etiquette guidelines
6. What are the benefits of open source development?
   A. Delay on bug fixes
   B. Poor security
   C. Size of contribution
   D. All of the above
   E. None of the above
7. What do most open source contributors expect to receive from their contribution?
   A. Money
   B. Recognition
   C. A paid job
   D. All of the above
   E. None of the above
8. Open source contributors should have one of these traits:
   A. Know how to code
   B. Own an IBM Model M keyboard
   C. Quick learner and willing to help others
   D. All of the above
   E. None of the above
9. Which of the following mediums of communication is popular among OSS communities for publishing Web articles or tutorials?
   A. Email
   B. IRC
   C. Internet blog
   D. Wiki
   E. None of the above
10. When engaging in discussions in an OSS project one must
    A. Doubt other people's capabilities
    B. Ignore those who shares different opinions
    C. Prove our point with reason and work

D. All of the above

E. None of the above

6

Chapter 6 – Starting your own open source project

In this chapter you will apply all the knowledge you have acquired thus far to experience how to start your own open source project. So think of something big!

In this chapter you will learn about:

- How to set up the basic infrastructure of your project.
- How to gather people for your community.
- How to promote your software and get more contributors.
- How to handle huge amounts of change requests from users.

6.1 Starting your own open source project: The big picture

Before starting an open source software development project, you must perform a self assessment test using the flow chart depicted in *Figure 6.1*.

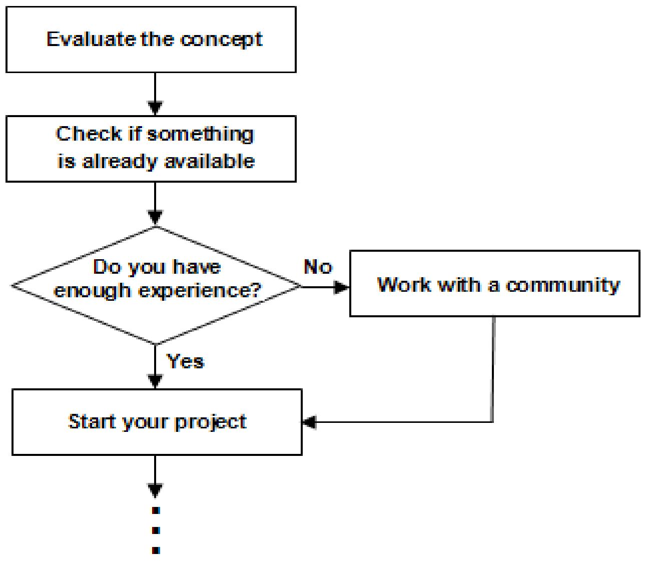

Evaluate the concept
Check if something is already available
Do you have enough experience?
No
Work with a community
Yes
Start your project

Figure 6.1 – Prerequisites for starting a new OSS project

As illustrated in *Figure 6.1*, the mandatory criteria for starting an OSS project is asking yourself two main questions:

1. What is the fundamental idea or requirement of your software? Do you feel many people would have same kind of requirements?
1. Is there any software already available in the market for the same purpose? If yes, does your product stand out for any reason? Why would people use it?

If you can convince yourself with the answers to the above questions, your project can be kicked off.

But, hold on. Do you have any prior experience contributing to an OSS project? Have you ever managed such a project? If your answer for either or both of these questions is NO, we suggest you be a part of at least one OSS project and understand their community culture. Get in touch with the management team and learn their experiences of dealing with various aspects of the project.

All sorted out? What are you waiting for then? Let’s get started with setting up the platform for the project.

6.2 Providing the ecosystem for your open source project

An ***ecosystem*** is a perfect environment, where interactive species work together as a single unit. In an OSS project, this environment undoubtedly refers to the OSS community.

As you may have guessed, the intention of this section is to find out how to build a community. This may sound as something simple, but gathering contributors together, and getting the work done, is easier said than done. To achieve this goal, you need to have management skills and expertise. This may be a good approach:

1. Establish a goal for your project.
2. Create a plan, roadmap or strategy to accomplish your goal.
3. Find community members.
4. Understand the basic needs of your project and community.
5. Arrange for necessary funds.

The only way to get members for your community is to raise awareness. Is there any person in your social network, who you feel can help in developing the software? Yes? Go, and get him convinced to contribute to your project. Other activities that will help are:

- Posting your project details in relevant forums.
- Building an official Web site. This site should convey all the details of your project. It should be the only portal from where users can access the code repository, issue tracking database, see the build status, browse the mailing lists, and subscribe to one or more of them. A Web site will allow you to reach contributors from all over the world.

- Getting the members to promote the project. Once you get a few members for your community, encourage all of them to promote your community. This makes the awareness effort multiplied by the number of members.

The first and most important thing that an OSS product and project needs is an appropriate license. When choosing the license, remember that users always prefer those ones which are well known in the industry; for example, BSD, GPL, etc. An unfamiliar license, even if it is very short, simple, and straight forward may discourage some users from choosing your software and contribute in the project. So, it is always recommended to select an unchanged version of a well known, popular open source license unless your product has special characteristics that would require specific terms and conditions. Refer to *Chapter 3 - Licensing*, to get more detailed guidance on the choice of a license.

Setting up channels of communication is next in the list. As discussed before, the easiest way to mass communicate all the details of your project is through a Web site. You also need to create separate mailing lists, IRC chat rooms, and blogs where people can discuss about specific matters of the project or product. Your active participation as well as the participation from other key community members will often encourage the newbies to get more involved into the project, because it shows that the project management and the main developers have the interest to understand your feedback, and fix problems proactively. Refer to *Chapter 5 - Participating in open source development*, for more information about this topic.

Other requirements of the project include various software development tools for version control, issue tracking, automating build process, designing, coding and testing. We have discussed about these tools in *Chapter 4 - Community driven development*.

6.3 Accepting contributions

Chapter 5 gave you an idea on how you can contribute to an open source project. This section tells you, how to drive the car on your own. Growth of an OSS project strongly relies on code or some other contributions from the outer world. And as you have already learned, the only way to gather such contributions in large scale is to make your software popular among the users, so that they feel enthusiastic about participating effectively in your project. It is not very easy to make a product widely visible in the present competitive market. However, if the key concept is innovative and if your software is distinctly ahead of other existing products, it is definitely not impossible!

OSS development projects typically start with small communities. Since commitment to the project is completely voluntary, some developers may not always devote much time on your project when they are busy with other important jobs. In such situations, it becomes really challenging to take the work forward with limited amount of workforce. On the other hand, once your community becomes famous and well known, you can naturally expect to get flooded with bug reports, patches, functionality improvement request etc. on a daily basis. But keep your head on your shoulder. Never try to handle all these requests yourself. Rather, delegate it properly. Redirect the issues and other reports or requests to the relevant development team, and ask any of the people within that group to address it.

When someone contributes code as part of a code change or new feature request, a developer should first spend some time understanding the requirement. If things are not clear, he can request the contributor to provide more details. Once the idea is clear to the developer, he needs to check the status of the present release cycle to ensure if it is the right time to accept the request. If it is, the developer should walk through the code, before allowing the user to check it in. If it's not a good time, the request should be scheduled for the next release.

Dealing with code reviews is painful when a developer has to do it all by himself. Make sure to delegate and involve more people into the review work. You can also let the experienced contributors whom you have worked with earlier and are entrusted by their quality of work, commit their work directly. Besides balancing the workload throughout the entire community, it also helps to clear the bottlenecks by reducing the number of pending user requests.

Another suggestion is to share with the community the criteria to accept code contributions. This way the community will know ahead of time what is required and code accordingly.

6.4 Exercises

1. Become a member of an existing OSS community and use your practical knowledge and experience while driving your own project.
2. Sourceforge provides an integrated platform for developing open source software. Explore http://sourceforge.net/ before starting your OSS project. You may like to make use of it.

6.5 Summary

In this chapter, you learned how to start an open source software development project of your own. You understood how to build up the basic infrastructure of your project, how to form a community, and how to satisfy its various needs.

Later, the chapter discussed about handling multiple aspects of accepting user contributions, and how to get more and more people involved in the project with effective contributions. You also learned the way to deal with many user change requests.

6.6 Review questions

1. Which two fundamental points a founder of an OSS project must evaluate before starting his project?
2. What are the basic steps to start your OSS project?
3. How can you grow the awareness of your project?
4. Why is it recommended to go for an unchanged version of an already existing OSS license?
5. How can you overcome the stress of the code review effort?

6. Which of the following is not a valid method for getting members for an OSS community?

   A. Face to face interaction.

   B. Posting project details in relevant forums.

   C. Creating an official website.

   D. Consulting a legal advisor.

   E. None of the above

7. Which of the following is not a mandatory requirement to start an OSS project?

   A. A license.

   B. A software development company's guidance.

   C. Communication channels.

   D. Software development tools.

   E. All of the above

8. What is the next step after receiving a feature enhancement request from a user?

   A. Requirement analysis.

   B. Checking the status of the release cycle.

   C. Asking the user to deliver the necessary code.

   D. Postponing the improvement until next release.

   E. None of the above

9. Which of the following methods should you follow to handle overflowing user change requests?

   A. Let it be a responsibility of one particular member of the community.

   B. Delegate authority and tasks.

   C. Don't accept new change requests, when many of such requests are pending.

   D. None of the above.

   E. All of the above

10. What is the best possible way to minimize the effort of screening user contributed code?

    A. Allow any user to commit their changes directly to the project branch without reviewing them.

    B. Ask every user to perform all kinds of testing of their own.

    C. Sharing the code selection criteria with the users.

D. Accept code only from known external users.

E. None of the above

7

Chapter 7 – Case Study: Contributing to an open source project

If you made it this far you must be thrilled and wanting to get your hands dirty. Use this chapter as your first guided tour to contribute to a real project. All the knowledge you've gained thus far can be applied following the step by step example provided in this chapter.

The subject of our study is the DB2 module to Ruby on Rails. Why? You may ask. Because the authors of this book are familiar with it and it will give us the opportunity to take a look to a very important tool.

7.1 Ruby on Rails and the DB2 module

There are a few tools that can help you develop Web applications, Ruby on Rails (RoR) is certainly one of the best. Ruby on Rails is a Web application framework developed by David Hansson and released in 2004, since then it has gained a lot of traction and has become widely used. Its adoption is much due to its simplicity and philosophy. Ruby on Rails builds on principles such as, Don't Repeat Yourself and Convention over Configuration. RoR is based on the Model-View-Controller (MVC) architecture and built-in tools such as scaffolding, which guarantee a rapid and easy development.

DB2 is a relational database management system, first released in 1983 by IBM. Even though it is not an open source software, DB2 has a no-charge version known as DB2 Express-C, giving us the ability to use one of the most high-performance relational databases available in the market for free!. DB2 can run on Linux, UNIX, Windows, Mac OS X (beta version at the time of writing), system i and z/OS® (mainframes). Its high availability and seamless stability has catered to the business crowd, but fear not, its features may well be what you are looking for. Like Ruby on Rails, DB2 builds on bold principles to succeed, DB2 supports for pureXML® technology which gives it a clear boost on application performance. It also provides the best compression support in the industry, allowing huge databases to be reduced to slightly more than half of their previous sizes.

Bringing together the benefits of these two technologies will put you at an obvious advantage. Here is where the DB2 Ruby on Rails module comes in. The DB2 Ruby on Rails module is the glue that makes this happen. A team of six developers ensures the project stays highly active and extremely well maintained, providing Rails developers with continuous integration.

Note:

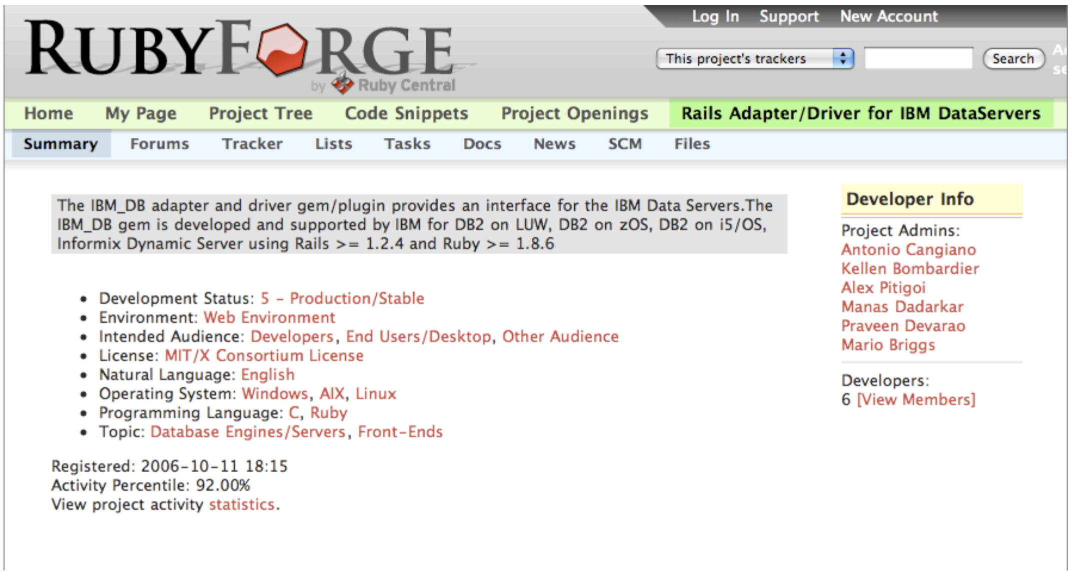
If you would like to learn more about Ruby on Rails, refer to the ebook *Getting started with Ruby on Rails*. For DB2, refer to the ebook *Getting started with DB2 Express-C*, or review *Appendix B, Up and Running with DB2*.

Both ebooks are free and part of the DB2 on Campus book series at ibm.com/db2/books.

7.2 The ruby forge

DB2 module for Ruby on Rails is hosted at ruby forge. A forge is basically a Web collaboration platform, providing developers with all the tools they need to manage the project and bring the discussion and code back to the community. A good forge will provide services like, mailing lists, source code management, wiki, bug tracking, forums and download archives. A forge holds a different number of projects, and some forges even have the ability to fork projects, enabling the user to start its own project from an existing one.

Figure 7.1 shows the ruby forge front page for the Ruby on Rails DB2 module.



The IBM_DB adapter and driver gem/plugin provides an interface for the IBM Data Servers.The IBM_DB gem is developed and supported by IBM for DB2 on LUW, DB2 on zOS, DB2 on i5/OS, Informix Dynamic Server using Rails >= 1.2.4 and Ruby >= 1.8.6

- Development Status: 5 - Production/Stable
- Environment: Web Environment
- Intended Audience: Developers, End Users/Desktop, Other Audience
- License: MIT/X Consortium License
- Natural Language: English
- Operating System: Windows, AIX, Linux
- Programming Language: C, Ruby
- Topic: Database Engines/Servers, Front-Ends

Registered: 2006-10-11 18:15
Activity Percentile: 92.00%
View project activity statistics.

Developer Info

Project Admins:
Antonio Cangiano
Kellen Bombardier
Alex Pitigoi
Manas Dadarkar
Praveen Devarao
Mario Briggs

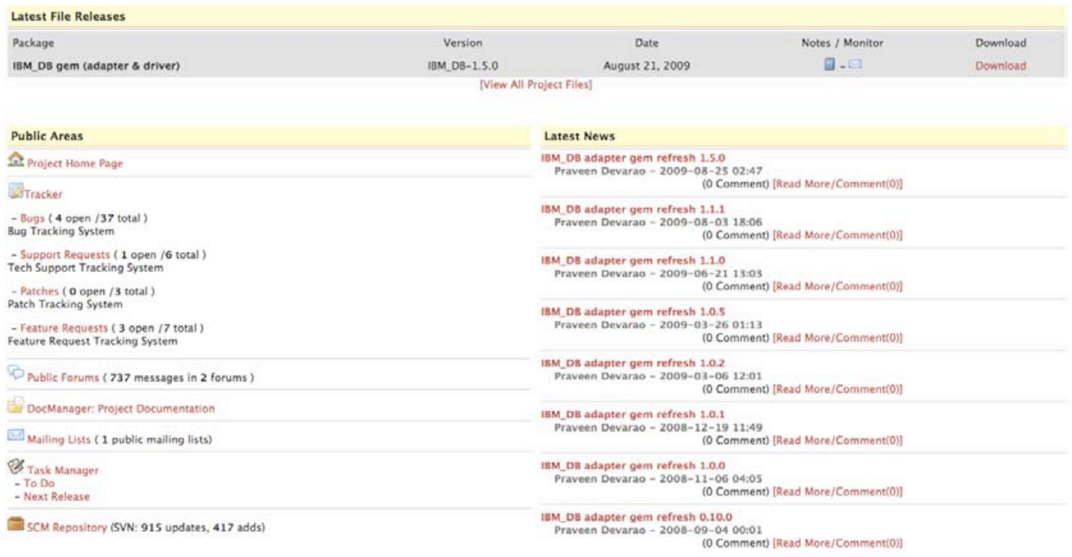
Developers:
6 [View Members]

Figure 7.1 - The RubyForge site showing the DB2 - RoR module main page (top)

This is the top section of the DB2 - RoR module page, which consists of a quick project description, a list of the project developers and some menus on the horizontal bar. These menus give you direct access to different sections of the Web page, containing the tools we have talked through the book. The main menu bar includes the following (listed from left to right):

- **Summary**: Provides a summary of the project. This is what you can see in *Figure 7.1*
- **Forums**: A place where users can exchange their ideas.
- **Tracker**: Takes you to the bug reporting tool with several but similar options.
- **Lists**: Provides a listing of several mailing lists available with capability to view, subscribe and unsubscribe.
- **Tasks**: Provides a todo list that project developers use to organize their activities.
- **Docs**: For documentation
- **News** and the **SCM** area: Gives you instructions to check out the project through subversion
- **Files**: Takes you to the download area containing all different versions of the Rails DB2 driver package in different formats.

Figure 7.2 shows the bottom part of the ruby forge DB2 - RoR module page.



Latest File Releases

Package	Version	Date	Notes / Monitor	Download
IBM_DB gem (adapter & driver)	IBM_DB-1.5.0	August 21, 2009	-	Download

[View All Project Files]

Public Areas

Project Home Page

Tracker

- Bugs (4 open /37 total)
Bug Tracking System

- Support Requests (1 open /6 total)
Tech Support Tracking System

- Patches (0 open /3 total)
Patch Tracking System

- Feature Requests (3 open /7 total)
Feature Request Tracking System

Public Forums (737 messages in 2 forums)

DocManager: Project Documentation

Mailing Lists (1 public mailing lists)

Task Manager
- To Do
- Next Release

SCM Repository (SVN: 915 updates, 417 adds)

Latest News

IBM_DB adapter gem refresh 1.5.0
Praveen Devarao - 2009-08-25 02:47
(0 Comment) [Read More/Comment(0)]

IBM_DB adapter gem refresh 1.1.1
Praveen Devarao - 2009-08-03 18:06
(0 Comment) [Read More/Comment(0)]

IBM_DB adapter gem refresh 1.1.0
Praveen Devarao - 2009-06-21 13:03
(0 Comment) [Read More/Comment(0)]

IBM_DB adapter gem refresh 1.0.5
Praveen Devarao - 2009-03-26 01:13
(0 Comment) [Read More/Comment(0)]

IBM_DB adapter gem refresh 1.0.2
Praveen Devarao - 2009-03-06 12:01
(0 Comment) [Read More/Comment(0)]

IBM_DB adapter gem refresh 1.0.1
Praveen Devarao - 2008-12-19 11:49
(0 Comment) [Read More/Comment(0)]

IBM_DB adapter gem refresh 1.0.0
Praveen Devarao - 2008-11-06 04:05
(0 Comment) [Read More/Comment(0)]

IBM_DB adapter gem refresh 0.10.0
Praveen Devarao - 2008-09-04 00:01
(0 Comment) [Read More/Comment(0)]

Figure 7.2 - The RubyForge site showing the DB2 - RoR module main page (bottom)

The bottom part of the Web page has a more detailed view of the items in the main menu, for example, the Latest File Releases area, gives you a description of the latest version as well as the date it came out and a quick link for downloading it. Below, the “Public Areas” section and the “Latest News” section follow the same principle. Either using the menu or these sections will lead you to the same content. It is important though that when you first arrive to a system like this you take some time to understand it. The sheer quantity of information with unusual nomenclatures can have an overwhelming impact.

7.3 Submitting a bug

The purpose of this chapter is to effectively use the knowledge we gathered from the rest of the book. One way of doing so is to imagine a situation where you have to submit a bug in the DB2 rails driver. As you have seen in the previous chapters, there are many ways of contributing to open source without having to type in a line of code. Submitting a bug may come across as trivial, but in fact most open source developers deal with poorly submitted bugs regularly.

Bugs exist since the first piece of software was created. Some bugs are nastier than others and can make software unusable. Bug reporting intends to decrease the amount of bugs in an application, making life easier for all of the users. A good bug report consists of the following:

- A detailed description of the system in which the software is running
- The input the user was feeding to the application
- The error message
- The expected behavior

Bug reports must be objective and based on facts, guesses have no place here. Even if you have the best of intentions by providing these guesses, it is not uncommon for a developer to spend a lot more time looking for a bug in a place there isn’t one, so the best thing you can do is just to fill in the bug with as much information as you can, and leave the rest to the developers. They will prompt you for input if they need to do so.

Let’s imagine for a moment that you find a real bug and you intend to fill in a bug report on the DB2 rails driver ruby forge page. "Where do I start?" you may ask. Well, the process below may help answer this question:

1. Check if there is already a submission for that bug. Duplicate bug submissions are a waste of time to you and to the developers.
2. If you don’t have an account on ruby forge, create one. If you have an account, login, then go to the ruby forge DB2 rails adapter page and select *tracker* from the main menu, then click on *Bugs*. Click *Submit New* button right below the main menu.
3. Fill in the fields:

- Category – This field requires some knowledge on the project structure. It refers to different sections of the project. For example the project may have a separate section for the user interface. In this case, the Ruby-DB2 project has two sections, Adapter and Driver. The adapter sits on rails ActiveRecord, which is a component responsible for the Object Relational Mapping (ORM). The driver, on the other hand, is responsible for making rails talk to IBM DB2 database.
- Group – The version of the IBM DB2 adapter and driver you are running.
- Summary – A short description of the bug.
- Detailed Description – This should include all the information necessary for the developer to reproduce the bug. In a system running rails, it would be obvious to start with the rails version, followed by the DB2 adapter and driver versions as well. All the error messages printed or logged by rails should be included. If you were doing anything at the time of the crash, describe it in detail.
- Upload Files – If you have a patch, attach and upload it. Sometimes when the output of the log files takes up too much space on the screen, it is nice to store it in one or more files and upload.

4. Press Submit and off you go.

Submitting bugs is simple science and one the best ways to contribute to an open source project; however, most people provide poor bug descriptions due to lack of knowledge or proper documentation. In this chapter you have seen how to fill a bug report in a simple and effective manner. There will be times when despite all your efforts, you won’t make yourself clear. Be persistent and try to get your thoughts through clearly. The developers will appreciate it and you will have your software fixed sooner. Good Bug Hunting!

8

Chapter 8 - Case Study: A sourceForge project, Technology Explorer for IBM DB2

Some open source projects grow along predictable lines. Some start well but quickly lose momentum. Just look at the number of projects on sourceforge.net that have had no recent activity. Other projects grow organically in unpredictable directions. The Technology Explorer for IBM DB2 or TE for short (http://db2mc.sf.net) is a project that has been through a long evolution. The TE started like all good open source projects, as an itch that needed scratching.

8.1 What is the Technology Explorer for IBM DB2?

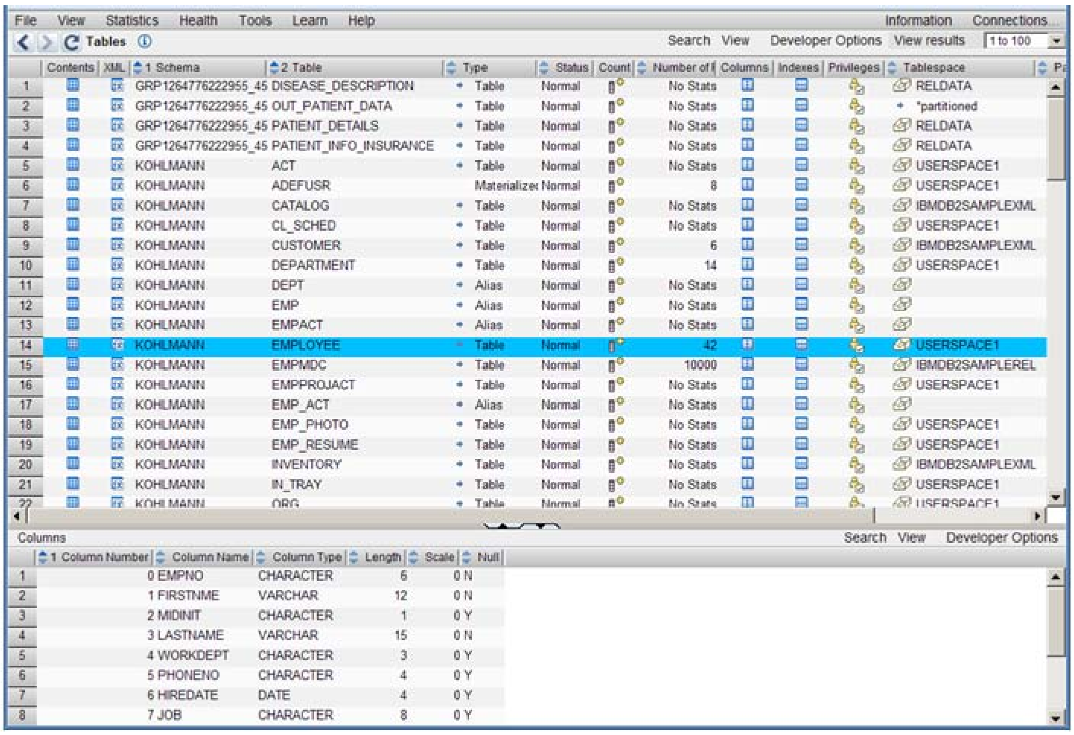
The Technology Explorer is a light weight, Web based console for DB2 for Linux, UNIX and Windows. The Technology Explorer is also a teaching and exploration tool for DB2. It is a great way to learn about DB2 because it doesn't just explain how to do things; it lets you play using a live database. The TE includes hundreds of ways to see inside the database and monitor what is going on. It also includes numerous tutorials and advanced techniques to get the most out of your system.

The Technology Explorer is meant to be a tool for the community. You can use it, change it, extend it and use it to share your ideas. Everything is defined using XML so you don't need to write code to make changes.

Some of the key features of the Technology Explorer are:

- Is a light weight Web based platform for interacting with DB2 for Linux, UNIX and Windows
- Is easy to expand and customize
- Works with DB2 for Linux, UNIX and Windows version 9.1, 9.5 and 9.7
- Connects to any DB2 data server using only an IP address
- Contains a wealth of content to highlight, demonstrate and teach you about DB2's core features

Figure 8.1 shows the Technology Explorer. It can run on most popular browsers.



	1 Schema	2 Table	Type	Status	Number of r	Tablespace
1	GRP1264776222955_45	DISEASE_DESCRIPTION	Table	Normal	No Stats	RELDATA
2	GRP1264776222955_45	OUT_PATIENT_DATA	Table	Normal	No Stats	*partitioned
3	GRP1264776222955_45	PATIENT_DETAILS	Table	Normal	No Stats	RELDATA
4	GRP1264776222955_45	PATIENT_INFO_INSURANCE	Table	Normal	No Stats	RELDATA
5	KOHLMANN	ACT	Table	Normal	No Stats	USERSPACE1
6	KOHLMANN	ADEFUSR	Materialize	Normal	8	USERSPACE1
7	KOHLMANN	CATALOG	Table	Normal	No Stats	IBMDB2SAMPLEXML
8	KOHLMANN	CL_SCHED	Table	Normal	No Stats	USERSPACE1
9	KOHLMANN	CUSTOMER	Table	Normal	6	IBMDB2SAMPLEXML
10	KOHLMANN	DEPARTMENT	Table	Normal	14	USERSPACE1
11	KOHLMANN	DEPT	Alias	Normal	No Stats	
12	KOHLMANN	EMP	Alias	Normal	No Stats	
13	KOHLMANN	EMPACT	Alias	Normal	No Stats	
14	KOHLMANN	EMPLOYEE	Table	Normal	42	USERSPACE1
15	KOHLMANN	EMPMDC	Table	Normal	10000	IBMDB2SAMPLEREL
16	KOHLMANN	EMPPROJACT	Table	Normal	No Stats	USERSPACE1
17	KOHLMANN	EMP_ACT	Alias	Normal	No Stats	
18	KOHLMANN	EMP_PHOTO	Table	Normal	No Stats	USERSPACE1
19	KOHLMANN	EMP_RESUME	Table	Normal	No Stats	USERSPACE1
20	KOHLMANN	INVENTORY	Table	Normal	No Stats	IBMDB2SAMPLEXML
21	KOHLMANN	IN_TRAY	Table	Normal	No Stats	USERSPACE1
22	KOHLMANN	ORG	Table	Normal	No Stats	USERSPACE1

Columns

	1 Column Number	Column Name	Column Type	Length	Scale	Null
1	0	EMPNO	CHARACTER	6	0	N
2	1	FIRSTNME	VARCHAR	12	0	N
3	2	MIDINIT	CHARACTER	1	0	Y
4	3	LASTNAME	VARCHAR	15	0	N
5	4	WORKDEPT	CHARACTER	3	0	Y
6	5	PHONENO	CHARACTER	4	0	Y
7	6	HIREDATE	DATE	4	0	Y
8	7	JOB	CHARACTER	8	0	Y

Figure 8.1 – The Technology Explorer for IBM DB2

8.2 A quick overview of the Technology Explorer for IBM DB2

This section provides a quick overview of the TE. You will learn the requirements to set it up, and its basic operations. The easiest way to get started wit the TE is by looking up its homepage on SourceForge http://sourceforge.net/projects/db2mc/

This site includes everything about the TE including setup instructions, wikis, requirements and forums as well as the source code.

8.2.1 Requirements for setting up the TE

To start working with the TE, you will need to:

- Install DB2
  - Download DB2 Express-C, the free version of DB2 from **http://www.ibm.com/db2/express/download.html**
- Install Apache
  - Download Apache Web server from **http://httpd.apache.org/**
- Install PHP
  - Download PHP from **http://www.php.net/**

- Install the IBM DB2 driver for PHP
    - Download this driver from **http://sourceforge.net/projects/db2mc/files/**
- Download and unzip TE
    - Download the TE from **http://sourceforge.net/projects/db2mc/**

For more information about DB2 refer to the free eBook *Getting Started with DB2 Express-C* that is part of the DB2 on Campus book series. Alternatively, you can also review *Appendix B, Up and Running with DB2*.

For instructions on how to set up Apache and PHP for TE refer also to the TE page on SourceForge. The development team is working on simplifying the set-up process. At the time you are reading this, a single package installer may already be available on the TE SourceForge site.

8.2.2 Some basic features and operations of the TE

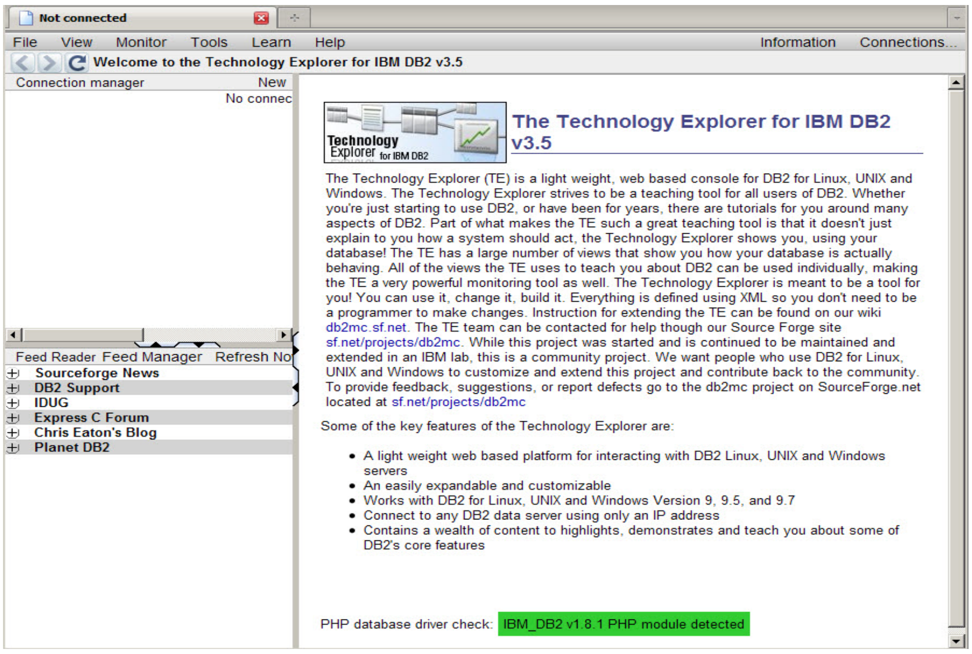
This document will briefly discuss some basic features and operations that the TE offers:

- Connect to a database
- Display table and view contents, and the output of queries
- Run ad hoc queries
- View and change database configuration and monitoring parameters
- Display basic monitoring information
- Create and work with tutorials and workshops

If you would like to know more about how to use TE in detail, a step-by-step guide is also available on the TE SourceForge site.

8.2.2.1 Connecting to a Database

Figure 8.2 shows the TE starting page. This is the page that appears when you start the TE in your browser.



The Technology Explorer for IBM DB2 v3.5

The Technology Explorer (TE) is a light weight, web based console for DB2 for Linux, UNIX and Windows. The Technology Explorer strives to be a teaching tool for all users of DB2. Whether you're just starting to use DB2, or have been for years, there are tutorials for you around many aspects of DB2. Part of what makes the TE such a great teaching tool is that it doesn't just explain to you how a system should act, the Technology Explorer shows you, using your database! The TE has a large number of views that show you how your database is actually behaving. All of the views the TE uses to teach you about DB2 can be used individually, making the TE a very powerful monitoring tool as well. The Technology Explorer is meant to be a tool for you! You can use it, change it, build it. Everything is defined using XML so you don't need to be a programmer to make changes. Instruction for extending the TE can be found on our wiki db2mc.sf.net. The TE team can be contacted for help though our Source Forge site sf.net/projects/db2mc. While this project was started and is continued to be maintained and extended in an IBM lab, this is a community project. We want people who use DB2 for Linux, UNIX and Windows to customize and extend this project and contribute back to the community. To provide feedback, suggestions, or report defects go to the db2mc project on SourceForge.net located at sf.net/projects/db2mc

Some of the key features of the Technology Explorer are:

- A light weight web based platform for interacting with DB2 Linux, UNIX and Windows servers
- An easily expandable and customizable
- Works with DB2 for Linux, UNIX and Windows Version 9, 9.5, and 9.7
- Connect to any DB2 data server using only an IP address
- Contains a wealth of content to highlights, demonstrates and teach you about some of DB2's core features

PHP database driver check: IBM_DB2 v1.8.1 PHP module detected

Figure 8.2 – TE starting page

The main menu bar is at the top of the page. The database connection manager is at the top left. The RSS feed reader is at the bottom left; and the introduction page is on the right.

To create a new database connection, click on *New* in the top right corner of the connection manager as shown in *Figure 8.3* below.

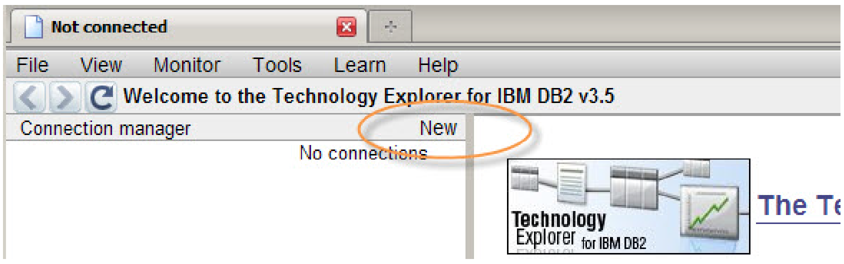
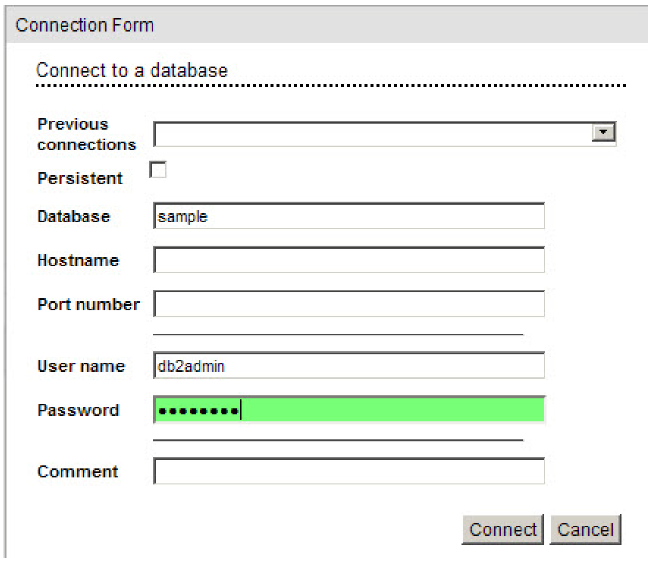


Figure 8.3 – Creating a new connection

If you are connecting to a database **on your local machine**, enter the database name, user name, and password, as shown in *Figure 8.4* below. The password field color will change from red to green when you enter a password.

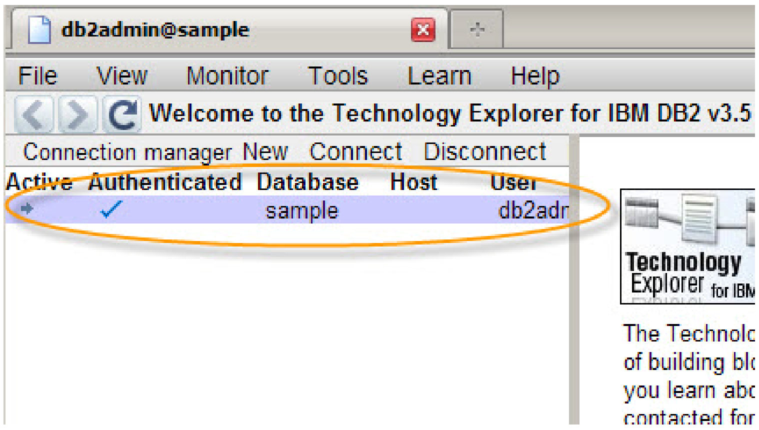

Connection Form

Connect to a database

Previous connections	
Persistent	☐
Database	sample
Hostname	
Port number	
User name	db2admin
Password	••••••••
Comment	

Figure 8.4 – Connection Form

If you are connecting to **a remote database**, then you must also enter the host name of the database machine as well as the port number used by the DB2 instance (the default is 50000). Click on *Connect*. The TE should load with the connection details in the top left panel, with an arrow under the *Active* field, and a check mark under the *Authenticated* field as shown in *Figure 8.5* below.



Active	Authenticated	Database	Host	User
➡	✓	sample		db2adr

The Technolo
of building bl
you learn abo
contacted for

Figure 8.5 – Connected to the Sample database

If you get an error message, check that:

- You have entered the correct combination of user name and password
- The user has authorization to connect to the database
- The database name is spelled properly
- For remote databases, make sure the host name and port number are correct
- The DB2 database server is running

Note:

Connecting to a DB2 database is discussed in detail in the free eBook *Getting started with DB2 Express-C.* You can also watch this video:

http://www.channeldb2.com/video/video/show?id=807741%3AVideo%3A4222

8.2.2.2 Display tables, views, and aliases

With the TE, you can display DB2 catalog tables, and views. Select *DB2 catalog views* from the *View* menu at the top of the page as illustrated in *Figure 8.6.*

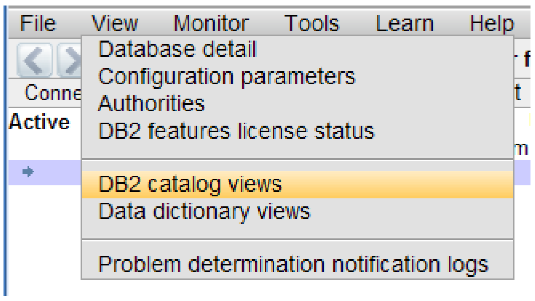
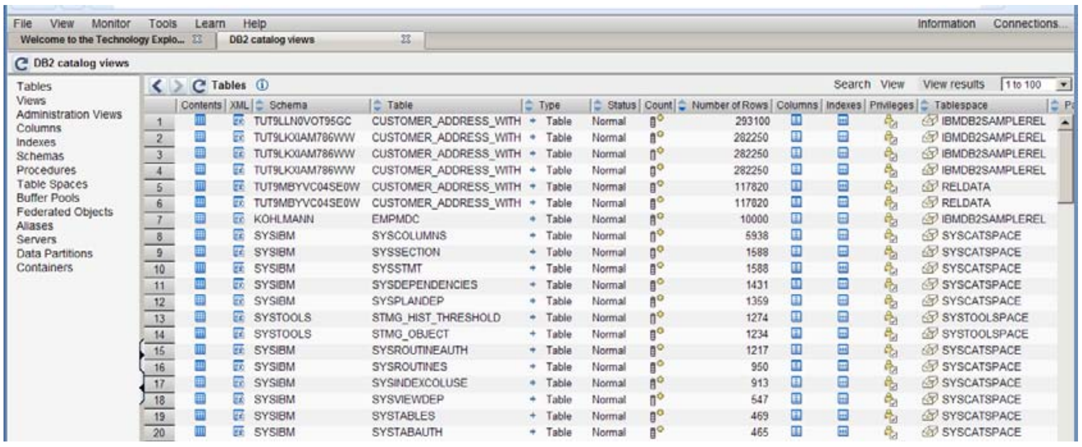


Figure 8.6 – Opening the Catalog Views

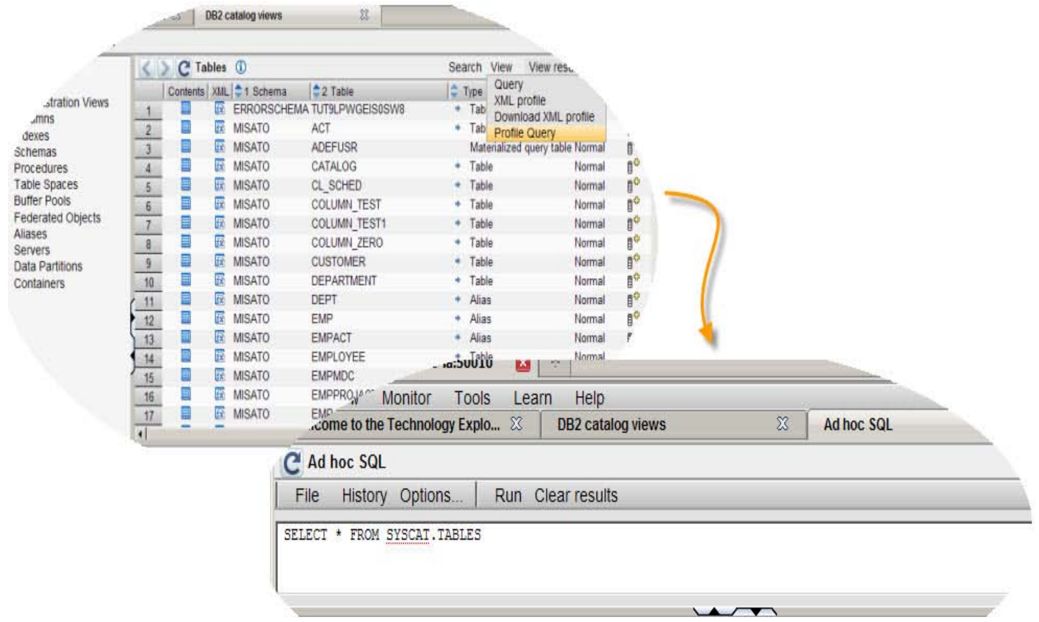
Figure 8.7 below provides an example of all items in the SYSCAT.TABLES view.



	Schema	Table	Type	Status	Number of Rows	Tablespace
1	TUT9LLN0VOT95GC	CUSTOMER_ADDRESS_WITH	Table	Normal	293100	IBMDB2SAMPLEREL
2	TUT9LKXIAM786WW	CUSTOMER_ADDRESS_WITH	Table	Normal	282250	IBMDB2SAMPLEREL
3	TUT9LKXIAM786WW	CUSTOMER_ADDRESS_WITH	Table	Normal	282250	IBMDB2SAMPLEREL
4	TUT9LKXIAM786WW	CUSTOMER_ADDRESS_WITH	Table	Normal	282250	IBMDB2SAMPLEREL
5	TUT9MBYVC04SE0W	CUSTOMER_ADDRESS_WITH	Table	Normal	117820	RELDATA
6	TUT9MBYVC04SE0W	CUSTOMER_ADDRESS_WITH	Table	Normal	117820	RELDATA
7	KOHLMANN	EMPMDC	Table	Normal	10000	IBMDB2SAMPLEREL
8	SYSIBM	SYSCOLUMNS	Table	Normal	5938	SYSCATSPACE
9	SYSIBM	SYSSECTION	Table	Normal	1588	SYSCATSPACE
10	SYSIBM	SYSSTMT	Table	Normal	1588	SYSCATSPACE
11	SYSIBM	SYSDEPENDENCIES	Table	Normal	1431	SYSCATSPACE
12	SYSIBM	SYSPLANDEP	Table	Normal	1359	SYSCATSPACE
13	SYSTOOLS	STMG_HIST_THRESHOLD	Table	Normal	1274	SYSTOOLSPACE
14	SYSTOOLS	STMG_OBJECT	Table	Normal	1234	SYSTOOLSPACE
15	SYSIBM	SYSROUTINEAUTH	Table	Normal	1217	SYSCATSPACE
16	SYSIBM	SYSROUTINES	Table	Normal	950	SYSCATSPACE
17	SYSIBM	SYSINDEXCOLUSE	Table	Normal	913	SYSCATSPACE
18	SYSIBM	SYSVIEWDEP	Table	Normal	547	SYSCATSPACE
19	SYSIBM	SYSTABLES	Table	Normal	469	SYSCATSPACE
20	SYSIBM	SYSTABAUTH	Table	Normal	465	SYSCATSPACE

Figure 8.7 – SYSCAT.TABLES catalog view

From this page you can see the query that retrieved the data in the table by clicking on *View -> Profile Query* as shown in *Figure 8.8*.

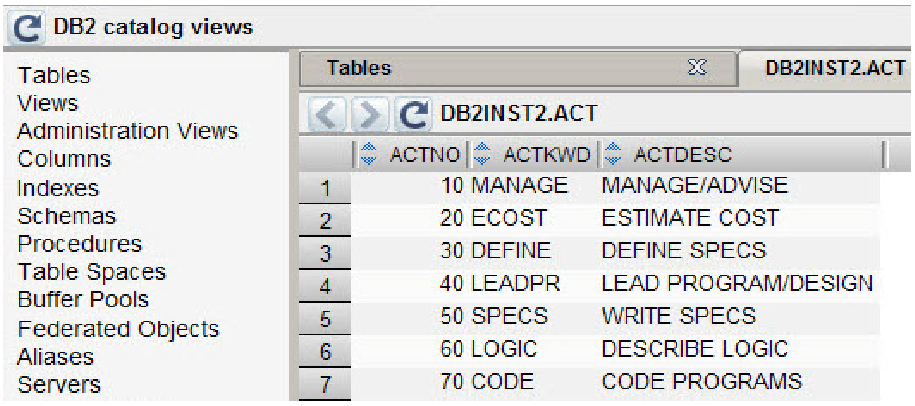


	Contents	XML	1 Schema	2 Table	Type	
1			ERRORSCHEMA	TUT9LPWGEIS0SW8	Tab	
2			MISATO	ACT	Tab	
3			MISATO	ADEFUSR	Materialized query table	Normal
4			MISATO	CATALOG	Table	Normal
5			MISATO	CL_SCHED	Table	Normal
6			MISATO	COLUMN_TEST	Table	Normal
7			MISATO	COLUMN_TEST1	Table	Normal
8			MISATO	COLUMN_ZERO	Table	Normal
9			MISATO	CUSTOMER	Table	Normal
10			MISATO	DEPARTMENT	Table	Normal
11			MISATO	DEPT	Alias	Normal
12			MISATO	EMP	Alias	Normal
13			MISATO	EMPACT	Alias	Normal
14			MISATO	EMPLOYEE	Table	Normal
15			MISATO	EMPMDC		
16			MISATO	EMPPROJ		
17			MISATO	EMP		

```
SELECT * FROM SYSCAT.TABLES
```

Figure 8.8 – Viewing the query that retrieved the data displayed on the page

You can also see the contents of each table by selecting the *See the contents* icon at the left of the display. A new tab opens with the contents of the table or view you selected. This is illustrated in *Figure 8.9* below for table DB2INST2.ACT.



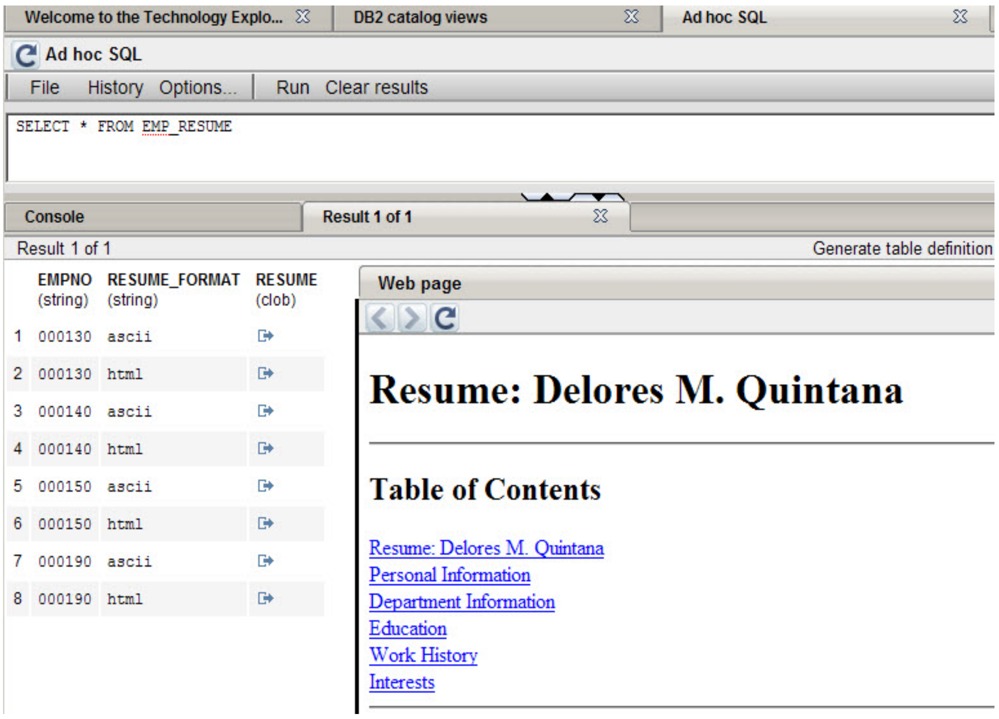
	ACTNO	ACTKWD	ACTDESC
1	10	MANAGE	MANAGE/ADVISE
2	20	ECOST	ESTIMATE COST
3	30	DEFINE	DEFINE SPECS
4	40	LEADPR	LEAD PROGRAM/DESIGN
5	50	SPECS	WRITE SPECS
6	60	LOGIC	DESCRIBE LOGIC
7	70	CODE	CODE PROGRAMS

Figure 8.9 – Contents of a table

8.2.2.3 Running ad hoc queries in TE

TE allows you to run SQL statements. You can see the results including BLOB, CLOB, DBCLOB, and XML data. *Figure 8.10* provides an example of an SQL statement (at the top) and the output (at the bottom). In the example the RESUME column is defined as

CLOB. On the left side of the result, there is an icon ⇨ that when clicked will display the CLOB contents on the right for the given employee.



```
SELECT * FROM EMP_RESUME
```

	EMPNO (string)	RESUME_FORMAT (string)	RESUME (clob)
1	000130	ascii	⇨
2	000130	html	⇨
3	000140	ascii	⇨
4	000140	html	⇨
5	000150	ascii	⇨
6	000150	html	⇨
7	000190	ascii	⇨
8	000190	html	⇨

Resume: Delores M. Quintana

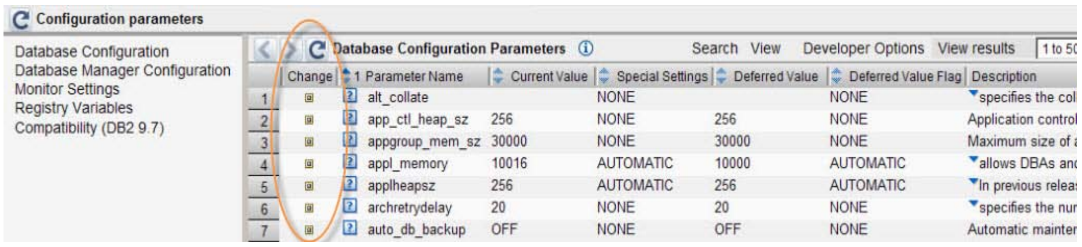
Table of Contents

Resume: Delores M. Quintana
Personal Information
Department Information
Education
Work History
Interests

Figure 8.10 – Running an ad hoc query

8.2.2.4 View and change database configuration and monitoring parameters

The TE allows you to change database configuration and monitoring parameters without having to revert to the command line. The icons highlighted in *Figure 8.11* below are used to change values.

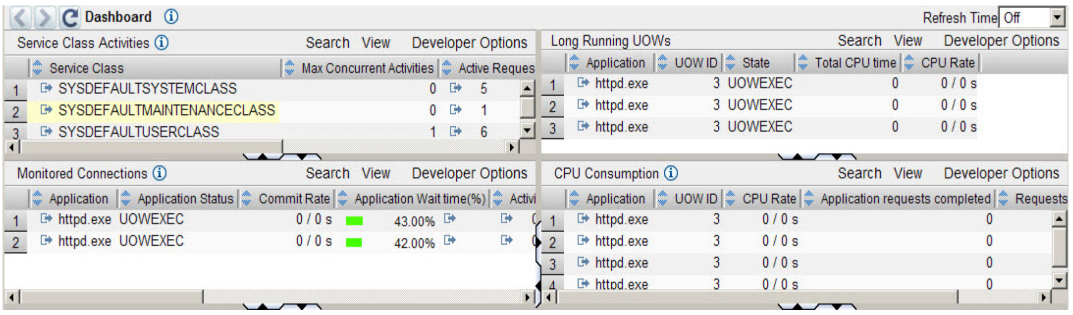


	Change	1 Parameter Name	Current Value	Special Settings	Deferred Value	Deferred Value Flag	Description
1	▣	alt_collate		NONE		NONE	specifies the col
2	▣	app_ctl_heap_sz	256	NONE	256	NONE	Application control
3	▣	appgroup_mem_sz	30000	NONE	30000	NONE	Maximum size of a
4	▣	appl_memory	10016	AUTOMATIC	10000	AUTOMATIC	allows DBAs and
5	▣	applheapsz	256	AUTOMATIC	256	AUTOMATIC	In previous releas
6	▣	archretrydelay	20	NONE	20	NONE	specifies the nur
7	▣	auto_db_backup	OFF	NONE	OFF	NONE	Automatic mainter

Figure 8.11 – Changing database configuration parameters

8.2.2.5 Display monitoring information

The TE includes monitoring information under the *Monitor* menu. DB2 can provide detailed monitoring information through SQL, which allows TE to show you what is happening in your DB2 server. *Figure 8.12* provides a sample of a monitoring dashboard. You can set it to refresh as often as you like.



Service Class Activities

	Service Class	Max Concurrent Activities	Active Reques
1	SYSDEFAULTSYSTEMCLASS	0	5
2	SYSDEFAULTMAINTENANCECLASS	0	1
3	SYSDEFAULTUSERCLASS	1	6

Long Running UOWs

	Application	UOW ID	State	Total CPU time	CPU Rate
1	httpd.exe	3	UOWEXEC	0	0 / 0 s
2	httpd.exe	3	UOWEXEC	0	0 / 0 s
3	httpd.exe	3	UOWEXEC	0	0 / 0 s

Monitored Connections

	Application	Application Status	Commit Rate	Application Wait time(%)	Activi
1	httpd.exe	UOWEXEC	0 / 0 s	🟩 43.00%	
2	httpd.exe	UOWEXEC	0 / 0 s	🟩 42.00%	

CPU Consumption

	Application	UOW ID	CPU Rate	Application requests completed	Requests
1	httpd.exe	3	0 / 0 s	0	
2	httpd.exe	3	0 / 0 s	0	
3	httpd.exe	3	0 / 0 s	0	
4	httpd.exe	3	0 / 0 s	0	

Figure 8.12 – The monitoring dashboard

8.2.2.6 Tutorials/Workshops

The TE offers many tutorials and workshops on DB2 as well as on the TE itself. *Figure 8.13* shows a small sample of the DB2 feature-based tutorials available. If you want to learn how to customize the TE, have a look at what is available under *Learn -> Workshops on Technology Explorer* menu.

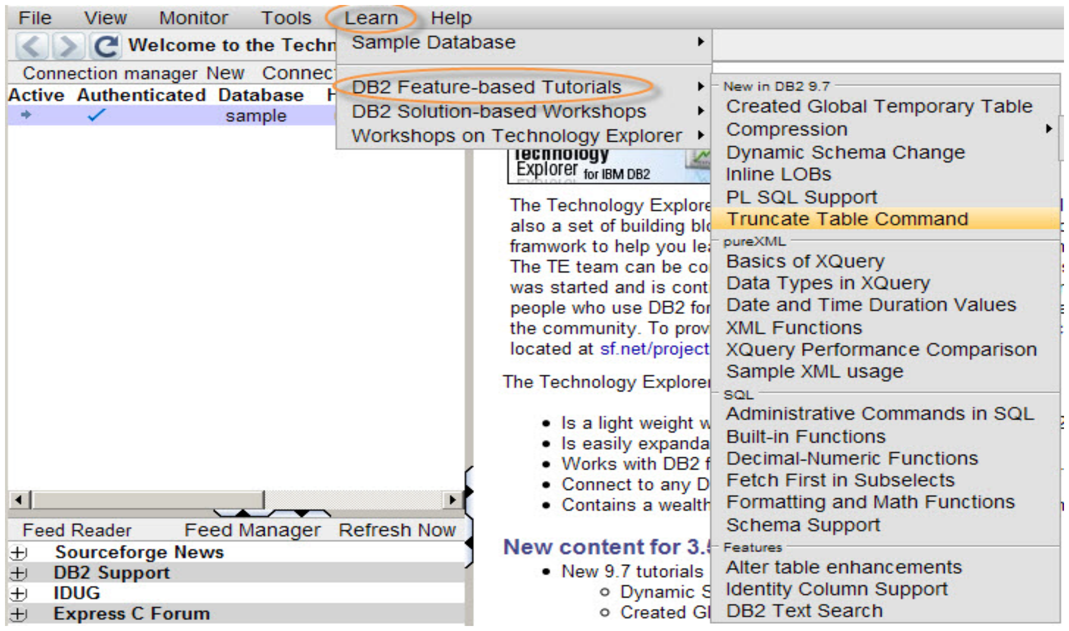


Figure 8.13 - Tutorials and workshops available with TE

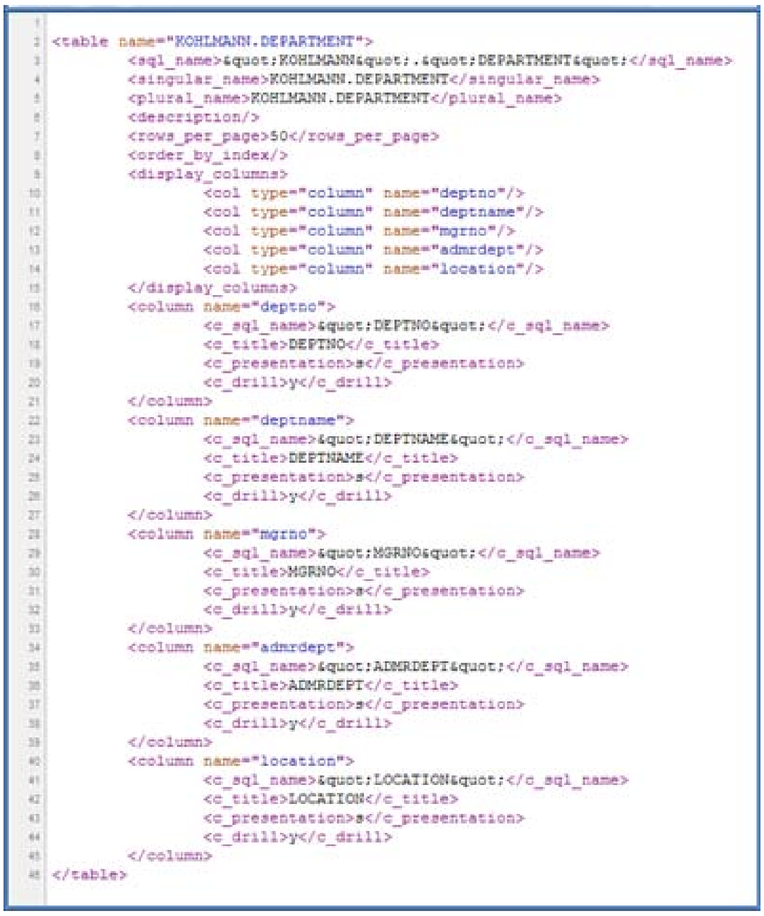
8.3 You need a key insight to build a project

The TE started as a hobby project and a way to explore DB2 and PHP. It started with basic information from a developerWorks® article (http://www.ibm.com/developerworks/data) on how to connect PHP to a DB2 data server. By writing a set of functions to encapsulate connection information and process SQL it was easier to quickly retrieve data from the server. Then by creating a set of PHP objects that describe tables and how to display them it became easy to quickly display data in a standard way. The first version was a sample PHP application for the sample database shipped with DB2. It demonstrated some basic principals of good PHP application development and examples of how to interface with DB2.

It evolved into an Open Source project when we understood the importance of flexibility in the design and the value of engaging the DB2 community. The first important insight was that sample data provided with the DB2 Sample database isn't as interesting as system data. The sample data serves no immediate practical purpose. We needed something with a practical application that would attract users to the project.

Fortunately DB2 9.1 has a ready source of data that is useful to anyone using a DB2 database - monitoring and performance data. By switching from a sample business application to a sample monitoring and management application the ***DB2 Monitoring Console Open Source Project*** was created (if you were wondering why the URL for the project is db2mc).

The second insight was that you could describe database tables and views using XML files. By switching to XML we created a far more flexible development environment. It became easy for anyone to customize the db2mc without writing a single line of code. We also took advantage of the outstanding XML support in IBM DB2. By using an XQuery statement we could automatically create a default *XML profile* for any table or view directly from the DB2 catalog with just a push of a button. Creating a new display used to take an hour or more. It now took minutes. *Figure 8.14* shows an example of an XML profile.



```

<table name="KOHLMANN.DEPARTMENT">
        <sql_name>&quot;KOHLMANN&quot;.&quot;DEPARTMENT&quot;</sql_name>
        <singular_name>KOHLMANN.DEPARTMENT</singular_name>
        <plural_name>KOHLMANN.DEPARTMENT</plural_name>
        <description/>
        <rows_per_page>50</rows_per_page>
        <order_by_index/>
        <display_columns>
                <col type="column" name="deptno"/>
                <col type="column" name="deptname"/>
                <col type="column" name="mgrno"/>
                <col type="column" name="admrdept"/>
                <col type="column" name="location"/>
        </display_columns>
        <column name="deptno">
                <c_sql_name>&quot;DEPTNO&quot;</c_sql_name>
                <c_title>DEPTNO</c_title>
                <c_presentation>s</c_presentation>
                <c_drill>y</c_drill>
        </column>
        <column name="deptname">
                <c_sql_name>&quot;DEPTNAME&quot;</c_sql_name>
                <c_title>DEPTNAME</c_title>
                <c_presentation>s</c_presentation>
                <c_drill>y</c_drill>
        </column>
        <column name="mgrno">
                <c_sql_name>&quot;MGRNO&quot;</c_sql_name>
                <c_title>MGRNO</c_title>
                <c_presentation>s</c_presentation>
                <c_drill>y</c_drill>
        </column>
        <column name="admrdept">
                <c_sql_name>&quot;ADMRDEPT&quot;</c_sql_name>
                <c_title>ADMRDEPT</c_title>
                <c_presentation>s</c_presentation>
                <c_drill>y</c_drill>
        </column>
        <column name="location">
                <c_sql_name>&quot;LOCATION&quot;</c_sql_name>
                <c_title>LOCATION</c_title>
                <c_presentation>s</c_presentation>
                <c_drill>y</c_drill>
        </column>
</table>
```

Figure 8.14 – XML Profile

This also added the ability to look at the contents of any table or view. Selecting the *contents* icons on the tables or view catalog page generates a default profile that is fed back to the page generator to create a default display for any page or view. This eliminated the tedious task of creating new pages.

With this new flexibility we also wanted to start engaging the DB2 community. Quick and simple displays are only useful if they contain meaningful information. The best way we felt to do this was to make the code available as an open source project. We hoped that DBA's would take some of the SQL queries they use every day and convert them to pages in the db2mc. More importantly we wanted them to contribute their work back to the DB2 community so other people could see the insight they gained from SQL that some DBA had taken years to develop.

The key insight that drives the Technology Explorer project is that to engage the community of database users you need an interface that is as flexible and dynamic as the SQL or XML languages. There are over 200 XML profiles shipped with the TE. They not only account for over 90% of the code shipped with the TE they represent an open language that people in the community can use to communicate their ideas of how they work.

8.4 You need to support and grow a community

Early adopters and contributors are the lifeblood of any open source project. You ignore them at your peril. For every 100 or 1000 users you are lucky to find one contributor. We were lucky enough that a very experienced and knowledgeable DBA found our project. Peter Prib (http://sourceforge.net/users/peterprib) has become an exceptional contributor to the project. He has added numerous new pages, new tools and utilities, and even more branches of code that we have yet been able to integrate into the mainline code stream. If he needs something to support his database administration consulting practice he adds it to the project. He has also been a great tester and regularly opens defects and adds requirements. Peter Prib's most important role is as our domain expert. He works with DB2 servers and customers every day. There is no substitute for that kind of practical experience. He doesn't just say that something is important, he invests his time.

We also receive contributions from universities and colleges. They are an outstanding resource for contribution. Working on an open source project not only gives a student practical experience but can become part of their portfolio of work. Most students work on software owned by another company. They can't show their code to anyone else or even prove that their code was used in the final product.

Contributing to an open source project has two advantages. First someone else usually approves your code and may give you valuable feedback on your style and technique. And second your code is available for anyone to see. A prospective employer can see for themselves that you have experience building code that was accepted by your peers.

The Technology Explorer for IBM DB2 uses a very open Apache license that makes it easy for anyone in the community to contribute. If you choose to contribute to a project, don't be surprised if they ask you to sign a contributor's agreement. They just need proof that the code you are contributing is original.

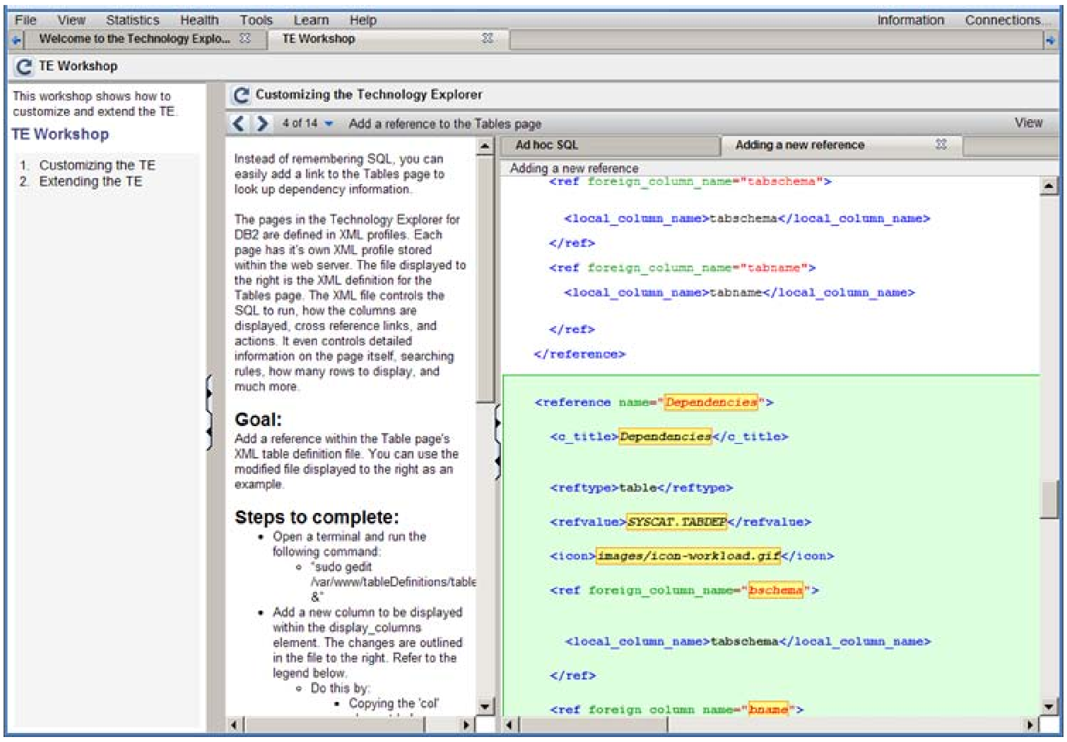
8.5 Make your project easy to adopt

Adoption isn't usability. It is much more than that. People go through a very well defined set of steps when they adopt new technology. As Jonathan Schwartz, from Sun Microsystems, has said, "*Adoption is a non-economic phenomena, no money is spent, only time…*" In other words, people spend time when they investigate new technology before they spend any money. Even if open source is free, if you want people to adopt your project, you need to make their initial time investment very low. You need to make your project convenient to use.

The first step is to ensure the software is easy to install and try out. If possible, build a live Web site where people can try out your project without installing anything. We made sure that we could set up the Technology Explorer for IBM DB2 so it could run in demonstration mode. Take a look at http://db2mc.dfw.ibm.com for an example. The setup ensures that anything destructive has been turned off.

Invest the time in an integrated installer. The only clearly negative feedback on the TE is that it takes too many steps to set up. A common problem with open source projects is that they are built on several components. The TE requires a Web server, DB2 PHP drivers as well as the TE code. For someone who has been through the process before, or is familiar with Web applications, it only takes five minutes, but to a novice, the number of steps may seem complex. We are finalizing an integrated installer that lays down all the components along with a single desktop icon to launch the TE and hope to have legal approval to ship all the components by the time you read this.

The second step is to ensure that someone can quickly see how to adapt the software to their own needs. In the more recent versions of the Technology Explorer we included demonstrations on how to customize the system including detailed examples. *Figure 8.15* illustrates a tutorial about how to customize the TE.


File View Statistics Health Tools Learn Help Information Connections
Welcome to the Technology Explo...
TE Workshop
TE Workshop
Customizing the Technology Explorer
4 of 14
Add a reference to the Tables page
View
Ad hoc SQL
Adding a new reference

This workshop shows how to customize and extend the TE.

TE Workshop

1. Customizing the TE
2. Extending the TE

Instead of remembering SQL, you can easily add a link to the Tables page to look up dependency information.

The pages in the Technology Explorer for DB2 are defined in XML profiles. Each page has it's own XML profile stored within the web server. The file displayed to the right is the XML definition for the Tables page. The XML file controls the SQL to run, how the columns are displayed, cross reference links, and actions. It even controls detailed information on the page itself, searching rules, how many rows to display, and much more.

Goal:
Add a reference within the Table page's XML table definition file. You can use the modified file displayed to the right as an example.

Steps to complete:
- Open a terminal and run the following command:
  - "sudo gedit /var/www/tableDefinitions/table &"
- Add a new column to be displayed within the display_columns element. The changes are outlined in the file to the right. Refer to the legend below.
  - Do this by:
    - Copying the 'col'

Adding a new reference

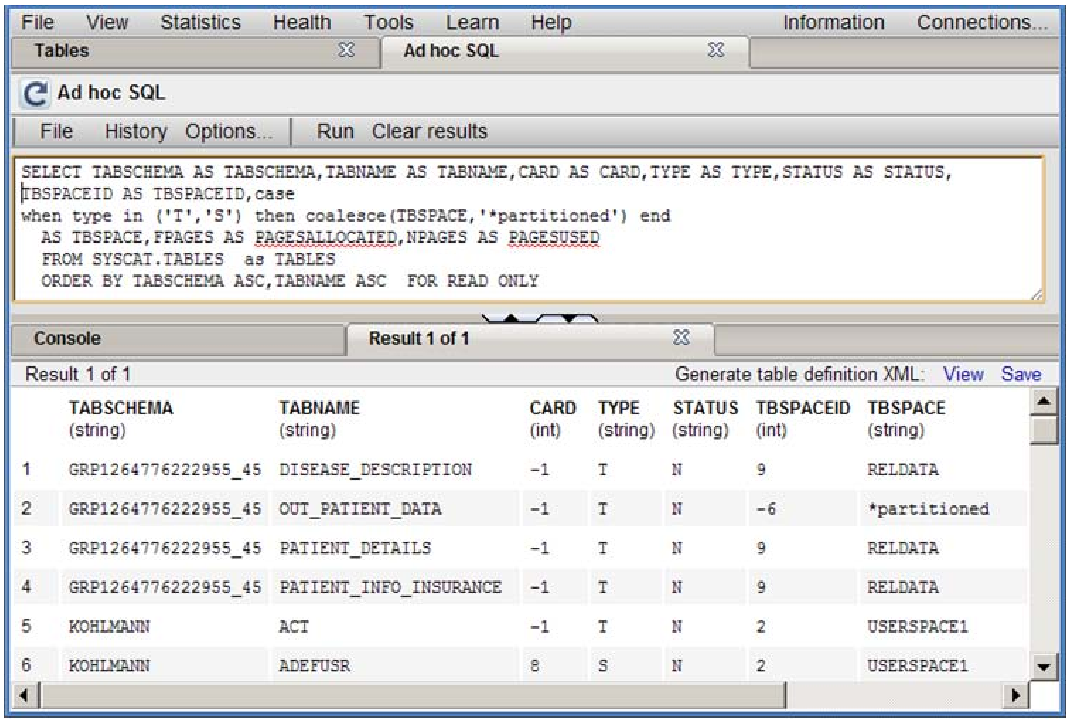
```
    <ref foreign_column_name="tabschema">

      <local_column_name>tabschema</local_column_name>
    </ref>
    <ref foreign_column_name="tabname">
      <local_column_name>tabname</local_column_name>

    </ref>
  </reference>

  <reference name="Dependencies">
    <c_title>Dependencies</c_title>

    <reftype>table</reftype>
    <refvalue>SYSCAT.TABDEP</refvalue>
    <icon>images/icon-workload.gif</icon>
    <ref foreign_column_name="bschema">

      <local_column_name>tabschema</local_column_name>
    </ref>
    <ref foreign column name="bname">
```

Figure 8.15 – Tutorial on how to customize the Technology Explorer

Adopting is also about easily making the transition from what you are already using. As mentioned before, you can easily create a new display in the Technology Explorer by selecting the XML icon from the table or view page. You can also export the SQL embedded in any XML profile to the ad hoc SQL page. Often we found that the results we needed were not available from a simple table or view. DB2 DBAs have also spent years developing SQL to answer key questions about the databases they are managing.

To give them a way to use their SQL we added the ad hoc SQL page. It not only allows them to reuse their SQL; they can quickly build new pages based on their existing SQL. The *Generate table definition XML* actions on the results tab will automatically generate an XML profile for any statement you can run in the ad hoc window. Building a bridge from existing practice to your new technology will greatly accelerate adoption.

Figure 8.16 illustrates running ad hoc statements in the TE.


File View Statistics Health Tools Learn Help Information Connections...
Tables
Ad hoc SQL
Ad hoc SQL
File History Options... Run Clear results
Console
Result 1 of 1
Result 1 of 1
Generate table definition XML: View Save

```
SELECT TABSCHEMA AS TABSCHEMA,TABNAME AS TABNAME,CARD AS CARD,TYPE AS TYPE,STATUS AS STATUS,
TBSPACEID AS TBSPACEID,case
when type in ('T','S') then coalesce(TBSPACE,'*partitioned') end
  AS TBSPACE,FPAGES AS PAGESALLOCATED,NPAGES AS PAGESUSED
  FROM SYSCAT.TABLES  as TABLES
  ORDER BY TABSCHEMA ASC,TABNAME ASC  FOR READ ONLY
```

	TABSCHEMA (string)	TABNAME (string)	CARD (int)	TYPE (string)	STATUS (string)	TBSPACEID (int)	TBSPACE (string)
1	GRP1264776222955_45	DISEASE_DESCRIPTION	-1	T	N	9	RELDATA
2	GRP1264776222955_45	OUT_PATIENT_DATA	-1	T	N	-6	*partitioned
3	GRP1264776222955_45	PATIENT_DETAILS	-1	T	N	9	RELDATA
4	GRP1264776222955_45	PATIENT_INFO_INSURANCE	-1	T	N	9	RELDATA
5	KOHLMANN	ACT	-1	T	N	2	USERSPACE1
6	KOHLMANN	ADEFUSR	8	S	N	2	USERSPACE1

Figure 8.16 – Running ad hoc statements

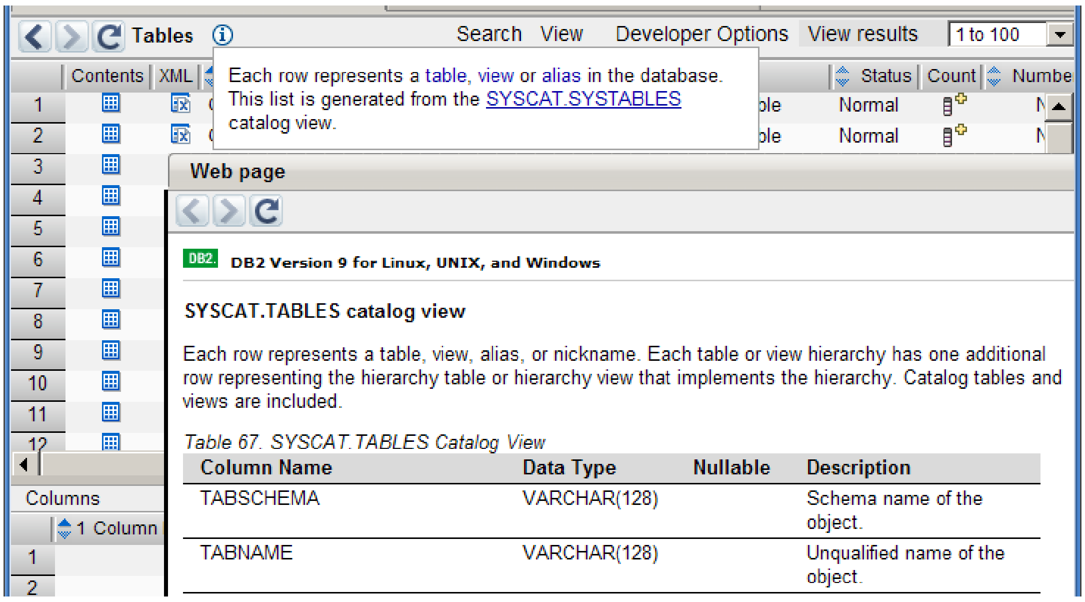
8.6 Understand your business model

A business model isn't always about making money by selling a product or service. Sometimes, especially with open source, it can be about saving money or sharing a cost and risk. It is always about understanding why you are investing in a project. In some cases open source is the basis for a services business. The software is free, but support comes at a cost.

Several companies are built around this model for Linux or PHP. In some cases there is a problem that is simply more cost effective for many people to share; Apache and Eclipse are good examples. In other cases it is about establishing a programming model or platform to deliver services, Google's Chrome for example.

In the case of the Technology Explorer for IBM DB2 we saw an opportunity to build a business model on demonstrations and tutorials. IBM was looking for ways to deliver more effective live tutorials and demonstrations for the DB2 data server while expanding their open user community and expanding their presence in universities and colleges. The flexible platform based on XML was the perfect starting point.

One of the first things we did to turn the TE from a sample program into a teaching tool was to integrate the DB2 Information Center (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7) into the program. Every page in the TE includes an information icon that explains where the information for the page comes from. *Figure 8.17* below provides an example where you can see that the table page is generated from the SYSCAT.TABLES catalog view. In the *Database Configuration Parameters* page we included information links to the Information Center for each parameter.



Each row represents a table, view or alias in the database. This list is generated from the SYSCAT.SYSTABLES catalog view.

DB2 Version 9 for Linux, UNIX, and Windows

SYSCAT.TABLES catalog view

Each row represents a table, view, alias, or nickname. Each table or view hierarchy has one additional row representing the hierarchy table or hierarchy view that implements the hierarchy. Catalog tables and views are included.

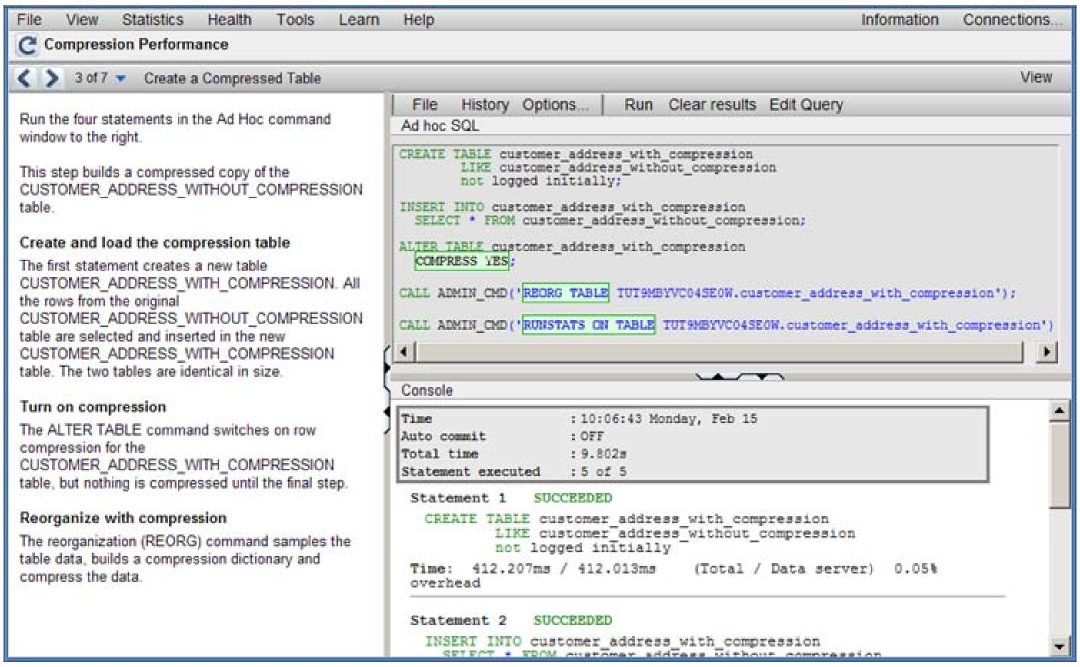
Table 67. SYSCAT.TABLES Catalog View

Column Name	Data Type	Nullable	Description
TABSCHEMA	VARCHAR(128)		Schema name of the object.
TABNAME	VARCHAR(128)		Unqualified name of the object.

Figure 8.17 –Integrated access to the DB2 Information Center

At the time we also had access to software and paper examples and tutorials. Many based on SQL examples. The biggest problem was that it was inconvenient for people to copy and paste the examples from an online source or worse to manually type from a paper during a workshop.

We extended the XML model to include step by step tutorials. Each tutorial included multiple steps each with a description to the left with a live ad hoc window and results screen to the right. Important parts of the SQL or XML can be highlighted in the ad hoc window. This is illustrated in *Figure 8.19*.


File View Statistics Health Tools Learn Help
Information Connections...
Compression Performance
3 of 7
Create a Compressed Table
View
File History Options... Run Clear results Edit Query
Ad hoc SQL
Console

Run the four statements in the Ad Hoc command window to the right.

This step builds a compressed copy of the CUSTOMER_ADDRESS_WITHOUT_COMPRESSION table.

Create and load the compression table

The first statement creates a new table CUSTOMER_ADDRESS_WITH_COMPRESSION. All the rows from the original CUSTOMER_ADDRESS_WITHOUT_COMPRESSION table are selected and inserted in the new CUSTOMER_ADDRESS_WITH_COMPRESSION table. The two tables are identical in size.

Turn on compression

The ALTER TABLE command switches on row compression for the CUSTOMER_ADDRESS_WITH_COMPRESSION table, but nothing is compressed until the final step.

Reorganize with compression

The reorganization (REORG) command samples the table data, builds a compression dictionary and compress the data.

```
CREATE TABLE customer_address_with_compression
        LIKE customer_address_without_compression
        not logged initially;

INSERT INTO customer_address_with_compression
  SELECT * FROM customer_address_without_compression;

ALTER TABLE customer_address_with_compression
  COMPRESS YES;

CALL ADMIN_CMD('REORG TABLE TUT9MBYVC04SE0W.customer_address_with_compression');

CALL ADMIN_CMD('RUNSTATS ON TABLE TUT9MBYVC04SE0W.customer_address_with_compression')
```

```
Time                    : 10:06:43 Monday, Feb 15
Auto commit             : OFF
Total time              : 9.802s
Statement executed      : 5 of 5

 Statement 1    SUCCEEDED
  CREATE TABLE customer_address_with_compression
          LIKE customer_address_without_compression
          not logged initially
 Time:  412.207ms / 412.013ms    (Total / Data server)  0.05%
 overhead

 Statement 2    SUCCEEDED
  INSERT INTO customer_address_with_compression
```

Figure 8.19 –Highlighting SQL

Once you have run through the tutorial, you are free to go back and alter the SQL or XML, and rerun the statements, all against a live database. We also support including pages from the DB2 Information Center or other pages, for example a list of tables, in the right side of the page instead of the ad hoc page.

As the number of tutorials grew, they were grouped together into larger workshops that focus on business problems. Additional functions like graphing and driving multiuser workloads were added to demonstrate more complex scenarios.

The XML language, which we developed to describe each tutorial, acts as a script. It is easy to modify and easy to copy to create a new tutorial. More importantly, just like the XML that defines the display pages in the TE, the tutorial XML acts as a language that people can use to describe their ideas. Instead of simply publishing documents, the tutorial structure can be used to explain a management process to a new DBA or highlight a valuable technique for getting the most out of DB2.

Make sure you take the time to understand why you are investing in an open source project. It may be as simple as a way to learn a new technology. With each step, the reason to keep investing time and resources may change. Whether it was as a sample PHP program, an open source monitor, or a community tool and teaching platform the TE project had a clear set of goals, stakeholders and investment. Throughout we have stayed focused on the original insight that started the project.

8.7 Keep your project current

Do everything in the open. Remember that source forge ranks projects based on community activity as much as it does by downloads. Even if you are working with a small group of contributors open a defect for every change. Track each improvement through a requirement and post your ideas to the forums. The more transparent your work the more active the project will appear and the better your ranking.

We have never advertised the TE. Most new users I talk to say they hear about it through word of mouth or through a lucky search on Google. Blog postings, webpage links, and other references bring most people to the Web site. Of the people who come to the TE sourceForge site about half stay and read about the project for at least an hour.

If you can find an insight to build a project around, make sure it is convenient to use and engages a larger community. Don't lose focus on why you are investing your time at each stage of the project and always do your work in the open.

A

Appendix A – Solutions to review questions

Chapter 1

1. In 1998 Eric Raymond and Bruce Perens first used the term open source in a conference in California.
2. FLOSS stands for Free Libre Open Source Software
3. Refer to section 1.4.1
4. Refer to section 1.4.2
5. The Mozilla Foundation.
6. C. KDE
7. A. Richard Stallman
8. D. A & B
9. B. Free as in "Free speech"
10. D. B & C

Chapter 2

1. MySQL.
2. Split Open Source software / commercial products.
3. POSS companies help the consumer manage the risks of deploying open source software.
4. Not limits on data storage and usage. No limits on type of hardware used. Paid support option
5. It is difficult to work in an OSS environment with many different OSS products that have not been tested together. Using the Platform providers' business model, a company offers a tested environment and provides support.
6. C. OS Providers is not a business model
7. A. Split open source / commercial products

8. D. All of the above
9. C. All of the above
10. D. All of the above

Chapter 3

1. Should be based on OSD (Open Source Definition).
2. To avoid possible lawsuits!
3. GPL, MIT, BSD
4. They have contradictory statements, however both satisfy the intentions for OSS licenses according to the OSI
5. Academic and reciprocal
6. Yes
7. Yes
8. Yes
9. False. Intellectual property (IP) is a superset containing copyrights, trademarks, etc.
10. E. None of the above

Chapter 4

1. In the software industry, “community driven development” broadly means an initiative under which a group of technologists work together with a common vision of producing open source software.
2. CVCS allows a single central repository for the entire project branch, which is shared by all the members of the community. In contrast, DVCS allows a developer or a tester to create his own code branch in addition to one or more shared central repository.
3. Abuky, Bugzilla, Buggit, BugRat, GNATS, JitterBug, and Mantis are a few examples of open source issue tracking tools.
4. The central item of an OSS package is the source code. Additionally, a standard package should also include a few accessories like installation guide, user manual, and other information which will help a newbie become familiar with the system. Packages of the software written in high level languages like C, C++ or something else that requires compilation of the source code, optionally may contain a pre-compiled executable version of the product ready to be installed into user’s computer.

5. Typical installation steps for open source software are – understanding the system configuration, compiling the source code and finally executing the installation file. Given below is the sequence of commands to accomplish these three steps in a UNIX® based system:

   ```
   # ./configure
   # make
   # make install
   ```

   The first command is executed from the location where the software package is extracted. It invokes a tiny program called configure (generally included within the package) to detect the hardware and software specifications of the machine and create a Makefile containing the set of instructions to complete the installation process. The make command compiles the source code of the software as dictated by the Makefile. And finally make install reads the configuration specifications of the computer from the Makefile and installs the software into it. Although the first two commands can be executed by any normal user, the last one needs root access to the system.
6. B. .zip is the valid format for Microsoft® Windows® packages. Whereas, .tar and .gz are specific to UNIX® platform.
7. A. Unified Modeling Language (UML) is a widely accepted standard used for software design purpose. It allows a designer to describe the structure, behavior, and interactions among different sub units of the system with the help of various simple diagrams.
8. C. 'myproduct' is the name of the software. '3', '4' and '5' are the "major", "minor" and "micro" components of the release number respectively.
9. D. A and C. Make and Ant are two build automation software. Whereas, Apache is an open source Web server and Darcs is a decentralized version control tool.
10. B. "Fixed" is not a state in the life cycle of an issue.

Chapter 5

1. It is ranked based on number of downloads, and community activity such as opening defects, checking in code, making feature requests, etc.
2. A leader performs functions that are not too different from a company's CEO:
   - He is the one making tough decisions
   - In big projects he is in charge of coordinating many of the project subdivisions.
3. In the early years of OSS, documentation was particularly important because working with OSS was not that straight forward. Only those who had the patience and skill could get this software working. Today, documentation is still very important since community members often rotate software development positions.

It is the responsibility of the documentation writer to ensure that every person can use the software in hand.

4. Translators are needed because OSS communities are worldwide communities. Though every member must have some English skills, having documentation in their own language will help.
5. Two guidelines can be:
    - Search before you ask.
    - Do not engage in acts of verbal aggression.
6. E
7. B
8. C
9. A
10. C

Chapter 6

1. Before starting an open source software development project, a person must evaluate following two key points:
    A. What is the fundamental idea or requirement of your software? Do you feel many people would have same kind of requirements?
    B. Is there any software already available in the market for the same purpose? If yes, where does your product stand out i.e. why would people use it?
2. Following are the basic steps for setting up the infrastructure of the project:
    A. Fix a definite mission or goal for your project.
    B. Plan a roadmap or strategy to accomplish your objective.
    C. Find people.
    D. Understand the basic needs of your project and community.
    E. Arrange for necessary funds.
3. Once you get a few members for your community, encourage all of them to publicize your community. This makes the awareness effort multiplied by the number of members.
4. Users always prefer licenses which are well known in the industry (e.g. BSD, GPL, etc.). An unfamiliar license, even if it is very short, simple, and straight forward may discourage some users from choosing your software and contribute in the project. So, it is always recommended to select an unchanged version of a well known,

popular open source license unless your product has an acute demand for very specific terms and conditions.

5. Dealing with code reviews becomes a pain when a developer has to do it everyday in bulk. Delegate: Involve more people into the review work, let the experienced users whom you have worked with earlier and are entrusted by their quality of work, to commit their work directly.
6. D. Legal advice is only needed while choosing appropriate license for the software. Others are valid options for gathering community members.
7. B. Open source software can be developed with or without any guidance of a software development company's guidance. All the other options are mandatory necessities of an OSS community.
8. A. Before a developer spends time going through the details of a code contribution or new feature request, he should try to understand its requirement. If there is any lack of clarity or consistency, he can request the user to provide more details. Once the idea is clear to the developer, he needs to check the status of the present release cycle to ensure if it is proper time to accept the request. If YES, the developer should ideally walk through the code, before permitting the user to check it in. And if the opposite is true, the request gets scheduled for the next release.
9. B. One should never try to handle all user change requests himself. Rather, he should delegate it properly, redirect the issues and other reports or requests to the relevant development team, and ask any of the people within that group to address it.
10. C. You may optionally share the code selection criteria of your community with the users; so that they keep them in mind while writing the codes and sending acceptance requests. This is the best possible way to minimize the effort of screening user contributed codes.

B

Appendix B – Up and running with DB2

This appendix is a good foundation for learning about DB2. This appendix is streamlined to help you get up and running with DB2 quickly and easily.

In this appendix you will learn about:

- DB2 packaging
- DB2 installation
- DB2 Tools
- The DB2 environment
- DB2 configuration
- Connecting to a database
- Basic sample programs
- DB2 documentation

Note:

For more information about DB2, refer to the free e-book *Getting Started with DB2 Express-C* that is part of this book series.

B.1 DB2: The big picture

DB2 is a data server that enables you to safely store and retrieve data. DB2 commands, XQuery statements, and SQL statements are used to interact with the DB2 server allowing you to create objects, and manipulate data in a secure environment. Different tools can be used to input these commands and statements as shown in *Figure B.1*. This figure provides an overview of DB2 and has been extracted from the *Getting Started with DB2 Express-C* e-book.

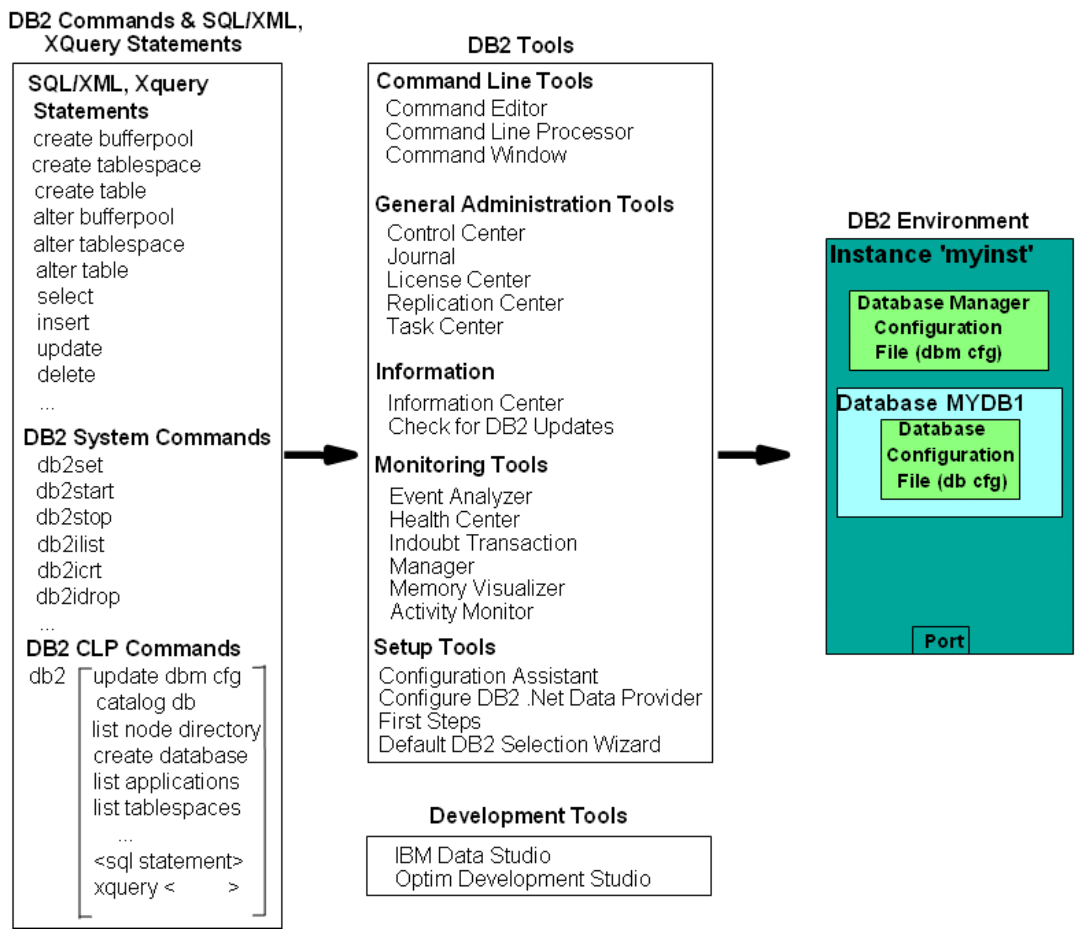


Figure B.1 - DB2 - The big picture

On the left-hand side of the figure, we provide examples of different commands and statements that users can issue. In the center of the figure, we list some of the tools where you can input these commands and statements, and on the right-hand side of the figure you can see the DB2 environment; where your databases are stored. In subsequent sections, we discuss some of the elements of this figure in more detail.

B.2 DB2 packaging

DB2 servers, clients and drivers are created using the same core components, and then are packaged in a way that allows users to choose the functions they need for the right price. This section describes the different DB2 editions or product packages available.

B.2.1 DB2 servers

Figure B.2 provides an overview of the different DB2 data server editions that are available.

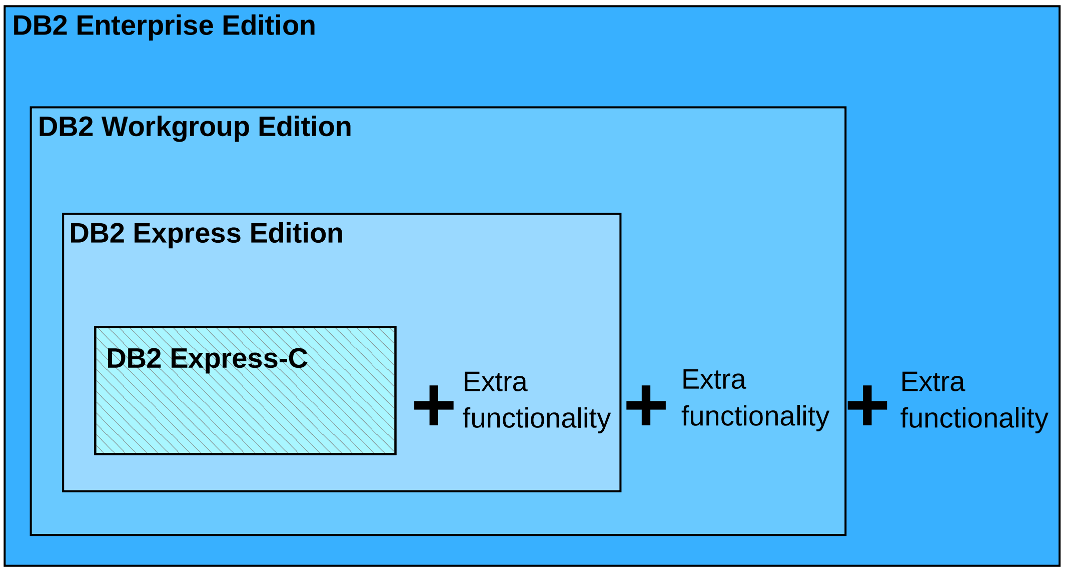


Figure B.2 - DB2 Server Packaging

As shown in *Figure B.2*, all DB2 server editions are built one on top of the other. DB2 Express-C is a free version of DB2, and it is the core component of all DB2 products. When additional functionality is added to DB2 Express-C, it becomes DB2 Express. Additional functionality added to DB2 Express, becomes DB2 Workgroup, and so on. *Figure B.2* illustrates why it is so easy to upgrade from DB2 Express-C to any other DB2 server should you need to in the future: *All DB2 servers editions are built based on DB2 Express-C*.

Also applications built for DB2 Express-C are applicable on other DB2 Editions as well. Your applications will function without any modifications required!

B.2.2 DB2 clients and drivers

When you install a DB2 server, a DB2 client component is also installed. If you only need to install a client, you can install either the IBM Data Server Client, or the IBM Data Server Runtime Client. *Figure B.3* illustrates these two clients.

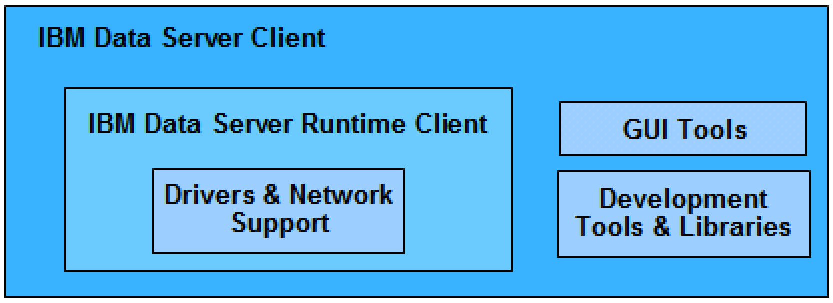

IBM Data Server Client
IBM Data Server Runtime Client
Drivers & Network Support
GUI Tools
Development Tools & Libraries

Figure B.3 - DB2 Clients

From the above figure, you can see the IBM Data Server Runtime client has all the components you need (driver and network support) to connect and work with a DB2 Data Server. The IBM Data Server client has this same support and also includes GUI Tools and libraries for application development.

In addition to these clients, provided are these other clients and drivers:

- DB2 Runtime Client Merge Modules for Windows: mainly used to embed a DB2 runtime client as part of a Windows application installation
- IBM Data Server Driver for JDBC and SQLJ: allows Java applications to connect to DB2 servers without having to install a client
- IBM Data Server Driver for ODBC and CLI: allows ODBC and CLI applications to connect to a DB2 server without having to install a client
- IBM Data Server Driver Package: includes a Windows-specific driver with support for .NET environments in addition to ODBC, CLI and open source. This driver was previously known as the IBM Data Server Driver for ODBC, CLI and .NET.

There is no charge to use DB2 clients or drivers.

B.3 Installing DB2

In this section we explain how to install DB2 using the DB2 setup wizard.

B.3.1 Installation on Windows

DB2 installation on Windows is straight-forward and requires the following basic steps:

1. Ensure you are using a local or domain user that is part of the Administrator group on the server where you are installing DB2.
2. After downloading and unzipping DB2 Express-C for Windows from ibm.com/db2/express, look for the file `setup.exe`, and double-click on it.
3. Follow the self- explanatory instructions from the wizard. Choosing default values is normally sufficient.
4. The following is performed by default during the installation:
   - DB2 is installed in `C:\Program Files\IBM\SQLLIB`
   - The DB2ADMNS and DB2USERS Windows operating system groups are created.
   - The instance **`DB2`** is created under `C:\Program Files\IBM\SQLLIB\DB2`
   - The DB2 Administration Server (DAS) is created
   - Installation logs are stored in:
     ```
     My Documents\DB2LOG\db2.log
     My Documents\DB2LOG\db2wi.log
     ```

- Several Windows services are created.

B.3.2 Installation on Linux

DB2 installation on Linux is straight-forward and requires the following basic steps:

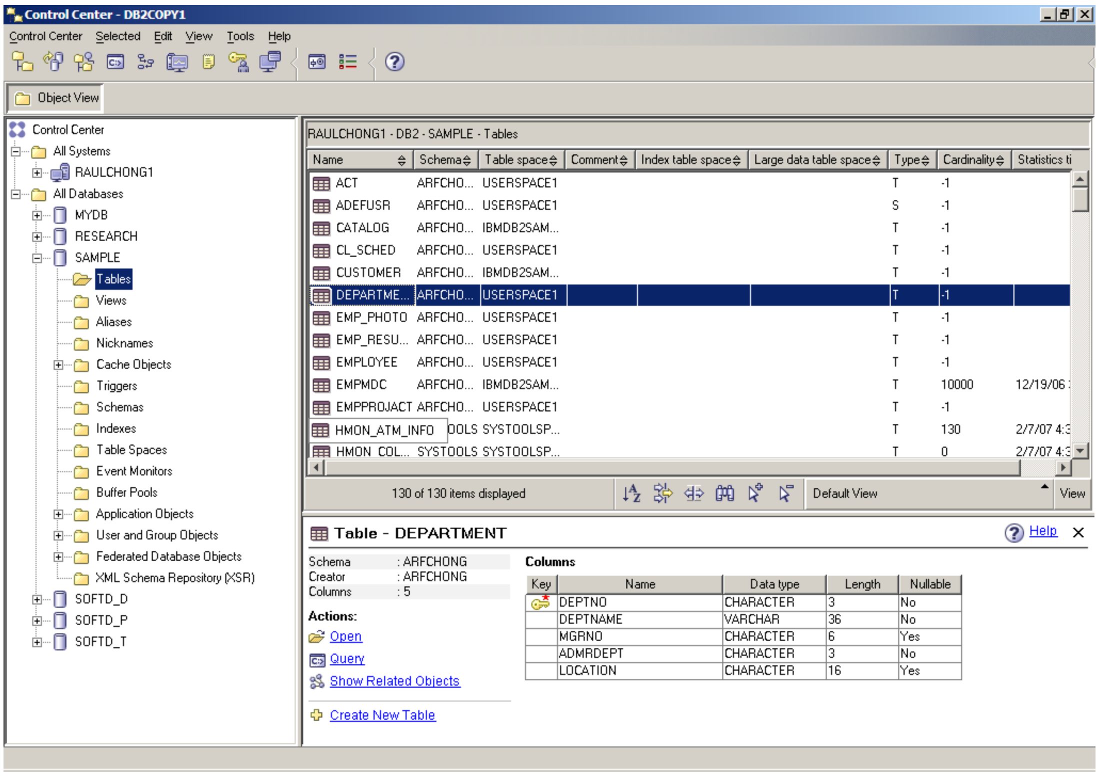
1. Log on as the Root user to install DB2.
2. After downloading DB2 Express-C for Linux from ibm.com/db2/express, look for the file `db2setup`, and execute it: **`./db2setup`**
3. Follow the self-explanatory instructions from the wizard. Choosing default values is normally sufficient.
4. The following is performed by default during installation:
   - DB2 is installed in `/opt/ibm/db2/V9.7`
   - Three user IDs are created. The default values are listed below:
     `db2inst1` (instance owner)
     `db2fenc1` (Fenced user for fenced routines)
     `dasusr1` (DAS user)
   - Three user groups are created corresponding to the above user IDs:
     `db2iadm1`
     `db2fadm1`
     `dasadm1`
   - Instance ***`db2inst1`*** is created
   - The DAS ***`dasusr1`*** is created
   - Installation logs are stored in:
     `/tmp/db2setup.his`
     `/tmp/db2setup.log`
     `/tmp/db2setup.err`

B.4 DB2 tools

There are several tools that are included with a DB2 data server such as the DB2 Control Center, the DB2 Command Editor, and so on. Starting with DB2 version 9.7 however; most of these tools are deprecated (that is, they are still supported but no longer enhanced) in favor of IBM Data Studio. IBM Data Studio is provided as a separate package not included with DB2. Refer to the ebook *Getting started with IBM Data Studio for DB2* for more details.

B.4.1 Control Center

Prior to DB2 9.7, the primary DB2 tool for database administration was the Control Center, as illustrated in *Figure B.4*. This tool is now deprecated, but still included with DB2 servers.



RAULCHONG1 - DB2 - SAMPLE - Tables

Name	Schema	Table space	Comment	Index table space	Large data table space	Type	Cardinality	Statistics ti
ACT	ARFCHO...	USERSPACE1				T	-1	
ADEFUSR	ARFCHO...	USERSPACE1				S	-1	
CATALOG	ARFCHO...	IBMDB2SAM...				T	-1	
CL_SCHED	ARFCHO...	USERSPACE1				T	-1	
CUSTOMER	ARFCHO...	IBMDB2SAM...				T	-1	
DEPARTME...	ARFCHO...	USERSPACE1				T	-1	
EMP_PHOTO	ARFCHO...	USERSPACE1				T	-1	
EMP_RESU...	ARFCHO...	USERSPACE1				T	-1	
EMPLOYEE	ARFCHO...	USERSPACE1				T	-1	
EMPMDC	ARFCHO...	IBMDB2SAM...				T	10000	12/19/06
EMPPROJACT	ARFCHO...	USERSPACE1				T	-1	
HMON_ATM_INFO	OOLS	SYSTOOLSP...				T	130	2/7/07 4:3
HMON_COL...	SYSTOOLS	SYSTOOLSP...				T	0	2/7/07 4:3

Table - DEPARTMENT

Schema	: ARFCHONG
Creator	: ARFCHONG
Columns	: 5

Columns

Key	Name	Data type	Length	Nullable
🔑	DEPTNO	CHARACTER	3	No
	DEPTNAME	VARCHAR	36	No
	MGRNO	CHARACTER	6	Yes
	ADMRDEPT	CHARACTER	3	No
	LOCATION	CHARACTER	16	Yes

Figure B.4 - The DB2 Control Center

To start the Control Center on Windows use *Start -> Programs -> IBM DB2 -> DB2COPY1 (Default) -> General Administration Tools -> Control Center* or alternatively, type the command **`db2cc`** from a Windows Command Prompt or Linux shell.

The Control Center is a centralized administration tool that allows you to:

- View your systems, instances, databases and database objects;
- Create, modify and manage databases and database objects;
- Launch other DB2 graphical tools

The pane on the left-hand side provides a visual hierarchy of the database objects on your system(s), providing a folder for Tables, Views, etc. When you double-click a folder (for example, the Tables folder, as shown in *Figure B.5*), the pane on the top right will list all of the related objects, in this case, all the tables associated with the **`SAMPLE`** database. If you select a given table in the top right pane, the bottom right pane provides more specific information about that table.

Right-clicking on the different folders or objects in the Object tree will bring up menus applicable to the given folder or object. For example, right-clicking on an instance and choosing *Configure parameters* would allow you to view and update the parameters at the instance level. Similarly, if you right-click on a database and choose *Configure parameters*, you would be able to view and update parameters at the database level.

B.4.2 Command Line Tools

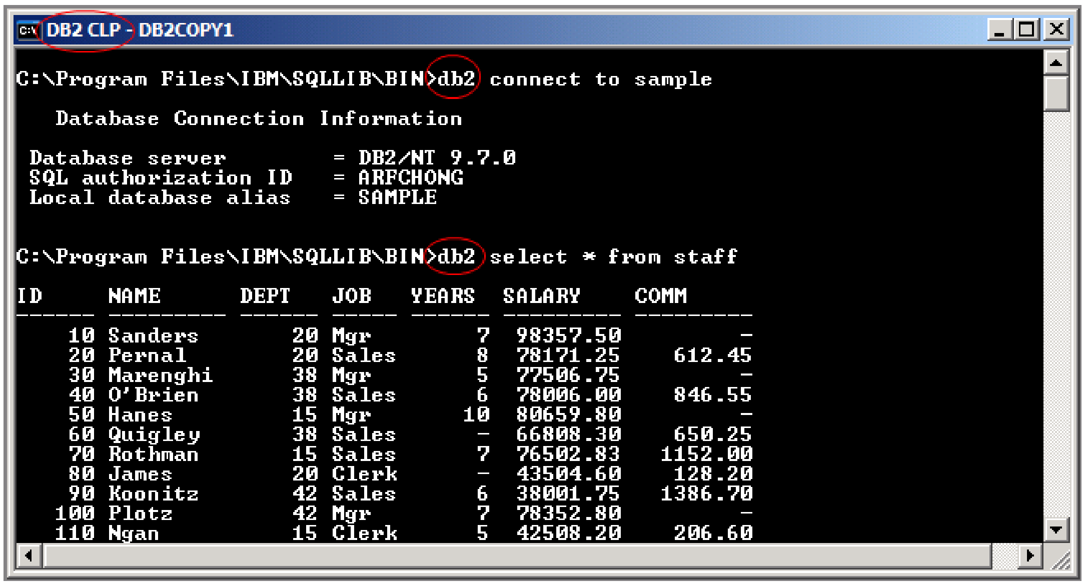
There are three types of Command Line tools:

- DB2 Command Window (only on Windows)
- DB2 Command Line Processor (DB2 CLP)
- DB2 Command Editor (GUI-based, and deprecated)

These tools are explained in more detail in the next sections.

B.4.2.1 DB2 Command Window

The DB2 Command Window is only available on Windows operating systems; it is often confused with Windows Command Prompt. Though they look the same, the DB2 Command Window, however, initializes the environment for you to work with DB2. To start this tool, use *Start -> Programs -> IBM DB2 -> DB2COPY1 (Default) -> Command Line Tools -> Command Window* or alternatively, type the command **db2cmd** from a Windows Command Prompt to launch it on another window. *Figure B.5* shows the DB2 Command Window.



```
C:\Program Files\IBM\SQLLIB\BIN>db2 connect to sample

   Database Connection Information

 Database server        = DB2/NT 9.7.0
 SQL authorization ID   = ARFCHONG
 Local database alias   = SAMPLE


C:\Program Files\IBM\SQLLIB\BIN>db2 select * from staff

ID     NAME      DEPT   JOB   YEARS  SALARY    COMM
------ --------- ------ ----- ------ --------- ---------
    10 Sanders       20 Mgr        7  98357.50         -
    20 Pernal        20 Sales      8  78171.25    612.45
    30 Marenghi      38 Mgr        5  77506.75         -
    40 O'Brien       38 Sales      6  78006.00    846.55
    50 Hanes         15 Mgr       10  80659.80         -
    60 Quigley       38 Sales      -  66808.30    650.25
    70 Rothman       15 Sales      7  76502.83   1152.00
    80 James         20 Clerk      -  43504.60    128.20
    90 Koonitz       42 Sales      6  38001.75   1386.70
   100 Plotz         42 Mgr        7  78352.80         -
   110 Ngan          15 Clerk      5  42508.20    206.60
```

Figure B.5 - The DB2 Command Window

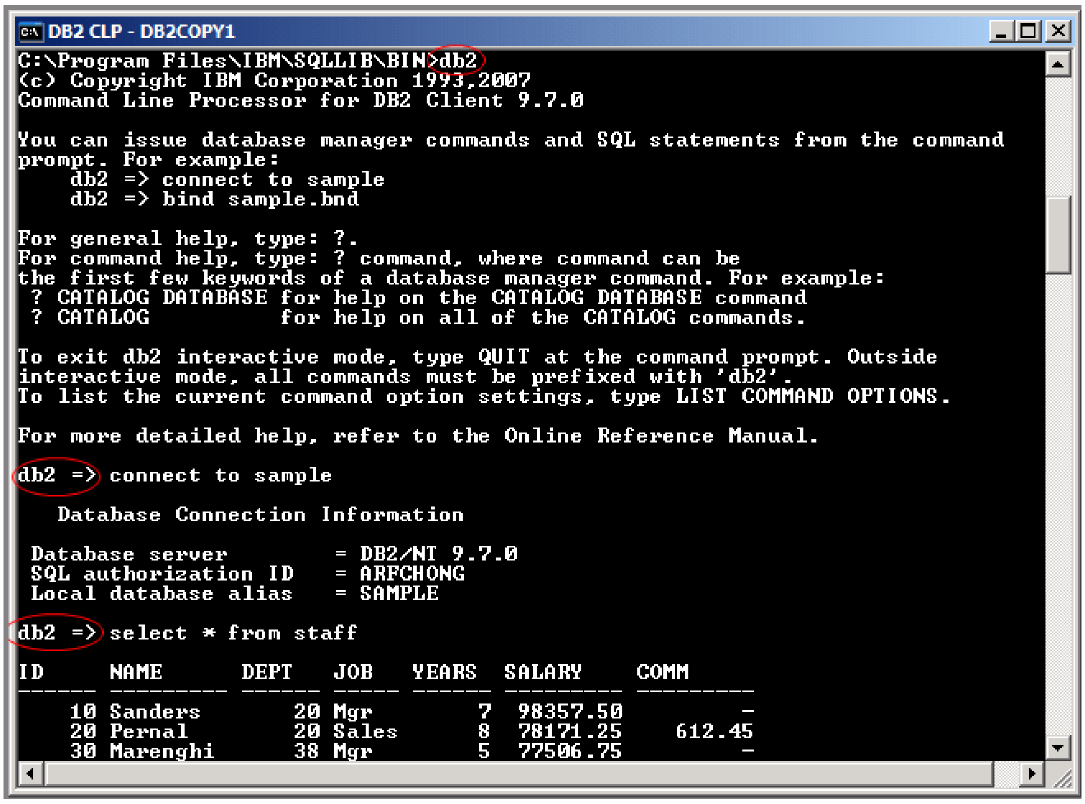
You can easily identify you are working in the DB2 Command Window by looking at the window title which always includes the words *DB2 CLP* as highlighted in the figure. From the DB2 Command Window, all commands must be prefixed with **db2**. For example, in the above figure, two statements are issued:

```
db2 connect to sample
db2 select * from staff
```

For Linux, the equivalent of the DB2 Command Window is simply the Linux shell (or terminal) where the DB2 environment has been set up by executing the `db2profile` file. This file is created by default and added to the `.login` file for the DB2 instance owner. By default the DB2 instance owner is ***`db2inst1`***.

B.4.2.2 DB2 Command Line Processor

The DB2 Command Line Processor (CLP) is the same as the DB2 Command Window, with one exception that the prompt is **db2=>** rather than an operating system prompt. To start the DB2 Command Line Processor on Windows, use *Start -> Programs -> IBM DB2 -> DB2COPY1 (Default) -> Command Line Tools -> Command Line Processor* or alternatively from a DB2 Command Window or Linux shell type **db2** and press *Enter*. The prompt will change to **db2** as shown in *Figure B.6*.



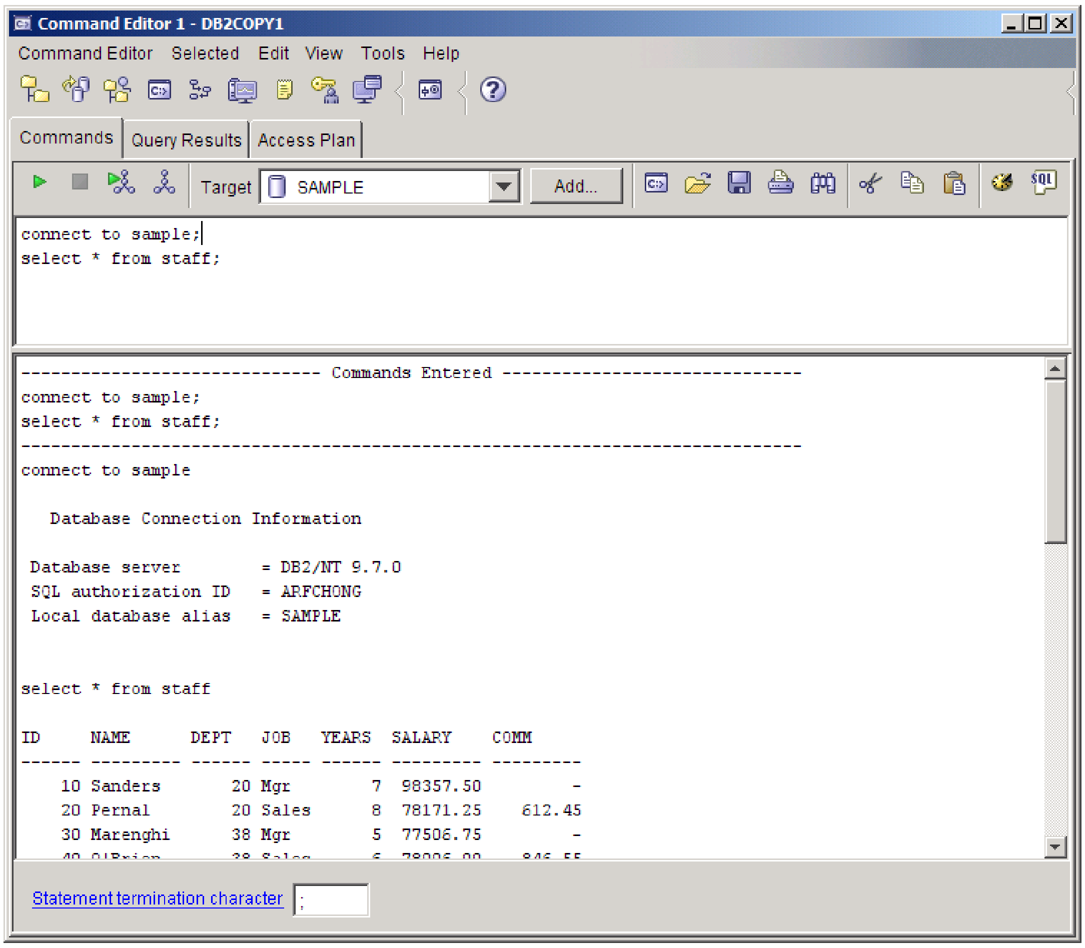
```
C:\Program Files\IBM\SQLLIB\BIN>db2
(c) Copyright IBM Corporation 1993,2007
Command Line Processor for DB2 Client 9.7.0

You can issue database manager commands and SQL statements from the command
prompt. For example:
    db2 => connect to sample
    db2 => bind sample.bnd

For general help, type: ?.
For command help, type: ? command, where command can be
the first few keywords of a database manager command. For example:
 ? CATALOG DATABASE for help on the CATALOG DATABASE command
 ? CATALOG          for help on all of the CATALOG commands.

To exit db2 interactive mode, type QUIT at the command prompt. Outside
interactive mode, all commands must be prefixed with 'db2'.
To list the current command option settings, type LIST COMMAND OPTIONS.

For more detailed help, refer to the Online Reference Manual.

db2 => connect to sample

   Database Connection Information

 Database server        = DB2/NT 9.7.0
 SQL authorization ID   = ARFCHONG
 Local database alias   = SAMPLE

db2 => select * from staff

ID     NAME      DEPT   JOB   YEARS  SALARY    COMM
------ --------- ------ ----- ------ --------- ---------
    10 Sanders       20 Mgr        7  98357.50         -
    20 Pernal        20 Sales      8  78171.25    612.45
    30 Marenghi      38 Mgr        5  77506.75         -
```

Figure B.6 - The DB2 Command Line Processor (CLP)

Note that *Figure B.6* also illustrates that when working in the CLP, you do not need to prefix commands with DB2. To exit from the CLP, type **`quit`**.

B.4.2.3 DB2 Command Editor

The DB2 Command Editor is the GUI version of the DB2 Command Window or DB2 Command Line Processor as shown in *Figure B.7*. This tool is deprecated for DB2 version 9.7.


Command Editor 1 - DB2COPY1
Command Editor Selected Edit View Tools Help
Commands
Query Results
Access Plan
Target
SAMPLE
Add...
Statement termination character

```
connect to sample;
select * from staff;
```

```
------------------------------ Commands Entered ------------------------------
connect to sample;
select * from staff;
------------------------------------------------------------------------------
connect to sample

   Database Connection Information

 Database server        = DB2/NT 9.7.0
 SQL authorization ID   = ARFCHONG
 Local database alias   = SAMPLE


select * from staff

ID     NAME      DEPT   JOB   YEARS  SALARY    COMM
------ --------- ------ ----- ------ --------- ---------
    10 Sanders       20 Mgr        7  98357.50         -
    20 Pernal        20 Sales      8  78171.25    612.45
    30 Marenghi      38 Mgr        5  77506.75         -
```

Figure B.7 - The DB2 Command Editor

B.5 The DB2 environment

Figure B.8 provides a quick overview of the DB2 environment.

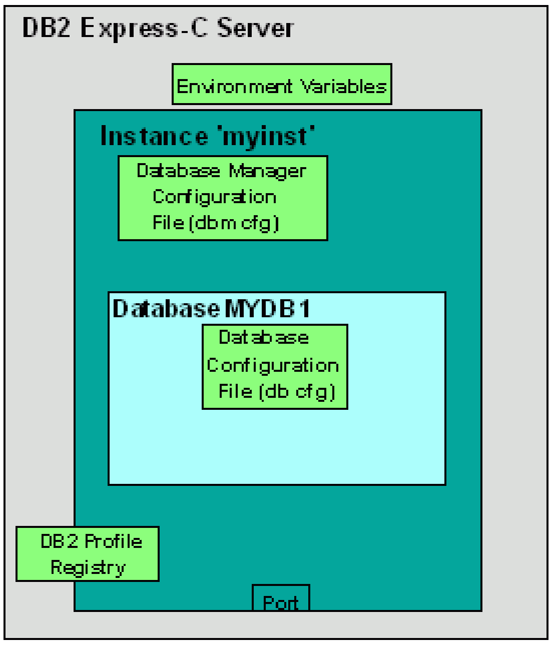


Figure B.8 - The DB2 Environment

The figure illustrates a server where DB2 Express-C has been installed. The smaller boxes in light green (Environment Variables, Database Manager Configuration File, Database Configuration File, DB2 Profile Registry) are the different areas where a DB2 server can be configured, and they will be explained in more detail in the next section. The larger dark green box represents an instance which in this example has the name **`myinst`**.

An ***instance*** is an environment where database objects can be created. On the same server, you can create several instances, each of which is treated independently. For example, you can use an instance for development, another one for test, and another one for production. *Table B.1* shows some useful commands you can use at the instance level. Note that the commands shown in this section can also be performed from DB2 GUI Tools.

Command	**Description**
db2start	Starts the current instance
db2stop	Stops the current instance
db2icrt <instance_name>	Creates a new instance
db2idrop <instance_name>	Drops an instance
db2ilist	Lists the instances you have on your system

db2 get instance	Lists the current active instance

Table B.1 - Useful instance-level DB2 commands

Within an instance you can create many databases. A ***database*** is a collection of objects such as tables, views, indexes, and so on. For example, in Figure B.8, the database ***`MYDB1`*** has been created within instance ***`myinst`***. *Table B.2* shows some commands you can use at the database level.

Command/SQL statement	**Description**
create database <database_name>	Creates a new database
drop database <database_name>	Drops a database
connect to <database_name>	Connects to a database
create table/create view/create index	SQL statements to create table, views, and indexes respectively

Table B.2 - Commands and SQL Statements at the database level

B.6 DB2 configuration

DB2 parameters can be configured using the Configuration Advisor GUI tool. The Configuration Advisor can be accessed through the Control Center by right clicking on a database and choosing *Configuration Advisor*. Based on your answers to some questions about your system resources and workload, the configuration advisor will provide a list of DB2 parameters that would operate optimally using the suggested values. If you would like more detail about DB2 configuration, keep reading. Otherwise, use the Configuration Advisor and you are ready to work with DB2!

A DB2 server can be configured at four different levels as shown earlier in *Figure B.8*:

- ***Environment variables*** are variables set at the operating system level. The main environment variable to be concerned about is DB2INSTANCE. This variable indicates the current instance you are working on, and for which your DB2 commands will apply.
- ***Database Manager Configuration File (dbm cfg)*** includes parameters that affect the instance and all the databases it contains. *Table B.3* shows some useful commands to manage the dbm cfg.

Command	**Description**
get dbm cfg	Retrieves information about the dbm cfg

update dbm cfg using <parameter_name> <value>	Updates the value of a dbm cfg parameter

Table B.3 - Commands to manipulate the dbm cfg

- ***Database Configuration File (db cfg)*** includes parameters that affect the particular database in question. *Table B.4* shows some useful commands to manage the db cfg.

Command	**Description**
get db cfg for <database_name>	Retrieves information about the db cfg for a given database
update db cfg for <database_name> using <parameter_name> <value>	Updates the value of a db cfg parameter

Table B.4 - Commands to manipulate the db cfg

- ***DB2 Profile Registry variables*** includes parameters that may be platform specific and can be set globally (affecting all instances), or at the instance level (affecting one particular instance). *Table B.5* shows some useful commands to manipulate the DB2 profile registry.

Command	**Description**
db2set -all	Lists all the DB2 profile registry variables that are set
db2set <parameter>=<value>	Sets a given parameter with a value

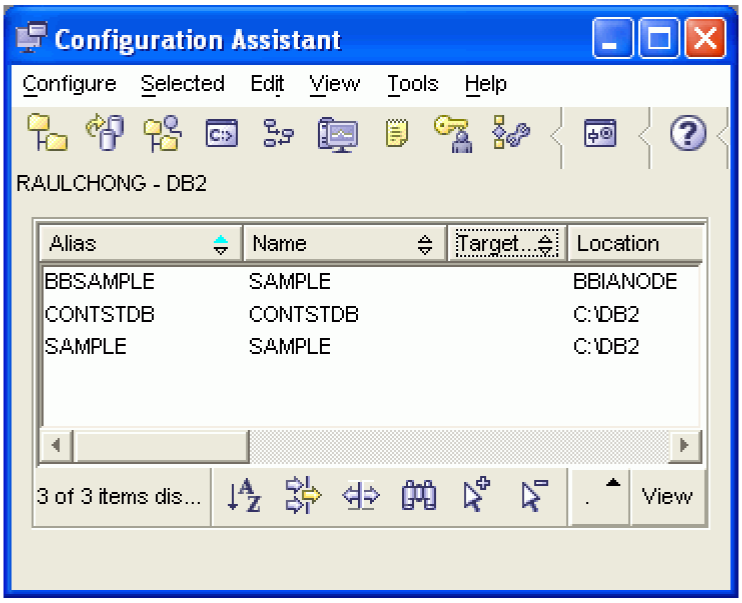
Table B.5 - Commands to manipulate the DB2 profile registry

B.7 Connecting to a database

If your database is local, that is, it resides on the same system where you are performing your database operation; the connection setup is performed automatically when the database is created. You can simply issue a `connect to database_name` statement to connect to the database.

If your database is remote, the simplest method to set up database connectivity is by using the Configuration Assistant GUI tool following these steps:

1. Start the Configuration Assistant from the system where you want to connect to the database. To start this tool, use the command **`db2ca`** from a Windows command prompt or Linux shell. *Figure B.9* shows the Configuration Assistant.



Alias	Name	Target...	Location
BBSAMPLE	SAMPLE		BBIANODE
CONTSTDB	CONTSTDB		C:\DB2
SAMPLE	SAMPLE		C:\DB2

Figure B.9 - The DB2 Configuration Assistant

2. From the Configuration Assistant, click on the *Selected --> Add database using Wizard* menu

3. From the *Select how you want to set up a connection* window, you can use *Search the network* if your network is small without many hubs. If you know the name of the server where DB2 resides, choose *Known systems* and drill down all the way to the database you want to connect. Proceed with the wizard using default values. If you do not know the name of your system, choose *Other systems* (*Search the network*). Note that this may take a long time if your network is large.

4. If *Search the network* does not work, go back to the *Select how you want to set up a connection* window, and choose *Manually configure a connection to a database*. Choose TCP/IP and click *next*. Input the *hostname or IP address* where your DB2 server resides. Input either the *service name or the port number*.

5. Continue with the wizard prompts and leave the default values.

6. After you finish your set up, a window will pop up asking you if you want to test your connection. You can also test the connection after the setup is finished by right-clicking on the database, and choosing *Test Connection*.

B.8 Basic sample programs

Depending on the programming language used, different syntax is required to connect to a DB2 database and perform operations. Below are links to basic sample programs which connect to a database, and retrieve one record. We suggest you first download (from ftp://ftp.software.ibm.com/software/data/db2/udb/db2express/samples.zip) all the sample programs in this section:

CLI program

http://www.ibm.com/developerworks/db2/library/techarticle/dm-0401chong/index.html#scenario1

ODBC program

http://www.ibm.com/developerworks/db2/library/techarticle/dm-0401chong/index.html#scenario2

C program with embedded SQL

http://www.ibm.com/developerworks/db2/library/techarticle/dm-0401chong/index.html#scenario3

JDBC program using Type 2 Universal (JCC) driver

http://www.ibm.com/developerworks/db2/library/techarticle/dm-0401chong/index.html#scenario6

JDBC program using Type 4 Universal (JCC) driver

http://www.ibm.com/developerworks/db2/library/techarticle/dm-0401chong/index.html#scenario8

Visual Basic and C++ ADO program - Using the IBM OLE DB provider for DB2 (IBMDADB2)

http://www.ibm.com/developerworks/db2/library/techarticle/dm-0402chong2/index.html#scenario1

Visual Basic and C++ ADO program - Using the Microsoft OLE DB Provider for ODBC (MSDASQL)

http://www.ibm.com/developerworks/db2/library/techarticle/dm-0402chong2/index.html#scenario2

Visual Basic and C# ADO.Net using the IBM DB2 .NET Data Provider

http://www.ibm.com/developerworks/db2/library/techarticle/dm-0402chong2/index.html#scenario3

Visual Basic and C# ADO.Net using the Microsoft OLE DB .NET Data Provider

http://www.ibm.com/developerworks/db2/library/techarticle/dm-0402chong2/index.html#scenario4

Visual Basic and C# ADO.Net using the Microsoft ODBC .NET Data Provider

http://www.ibm.com/developerworks/db2/library/techarticle/dm-0402chong2/index.html#scenario5

B.9 DB2 documentation

The DB2 Information Center provides the most up-to-date online DB2 documentation. The DB2 Information Center is a web application. You can access the DB2 Information Center online (http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp), or you can download and install the DB2 Information Center to your local computer. Links to the online DB2 Information Center as well as downloadable versions are available at http://www.ibm.com/software/data/db2/9/download.html?S_TACT=download&S_CMP=expcsite

References

[1] STREICHER, Martin. *Open source licensing, Part 1: The intent*, IBM developerWorks, 2005 http://www.ibm.com/developerworks/opensource/library/os-license/index.html

[2] BERCZUK, S. and APPLETON, B. *Software Configuration Management Patterns: Effective Teamwork, Practical Integration*. Addison-Wesley Professional, 2003

[3] BERCZUK, S., APPLETON, B. and KONIECZKA, S. *Build Management for an Agile Team.* CM Journal, Oct 2003 (Vol 2. No. 10)

[4] MYERS, G. *The Art of Software Testing.* John Wiley & Sons; Feb 1979

Resources

Web sites

1. DB2 Express-C web site:

   ibm.com/db2/express

   Use this web site to download the image for DB2 Express-C servers, DB2 clients, DB2 drivers, manuals, access to the team blog, mailing list sign up, etc.

2. DB2 Express-C forum:
   www.ibm.com/developerworks/forums/dw_forum.jsp?forum=805&cat=19

   Use the forum to post technical questions when you cannot find the answers in the manuals yourself.

3. DB2 Information Center

   http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp

   The information center provides access to the online manuals. It is the most up to date source of information.

4. developerWorks

   http://www-128.ibm.com/developerworks/db2

   This Web site is an excellent resource for developers and DBAs providing access to current articles, tutorials, etc. for free.

5. alphaWorks®

   http://www.alphaworks.ibm.com/

   This Web site provides direct access to IBM's emerging technology. It is a place where one can find the latest technologies from IBM Research.

6. planetDB2

   www.planetDB2.com

   This is a blog aggregator from many contributors who blog about DB2.

7. DB2 Technical Support

If you purchased the 12 months subscription license of DB2 Express-C, you can download fixpacks from this Web site.

http://www.ibm.com/software/data/db2/support/db2_9/

8. ChannelDB2

   ChannelDB2 is a social network for the DB2 community. It features content such as DB2 related videos, demos, podcasts, blogs, discussions, resources, etc. for Linux, UNIX, Windows, z/OS, and i5/OS®.

   http://www.ChannelDB2.com/

9. How to Ask Questions The Smart Way:

   http://catb.org/esr/faqs/smart-questions.html#intro

   A manual by Eric S. Raymond and Rick Moen who will enlighten new users in the art of getting technical answers online.

10. Ruby Language Site:

    http://www.ruby-lang.org

    Your starting point to the ruby community.

11. The Cathedral and the Bazaar

    An almost complete online version of the book by Eric S. Raymond, on the open source developing model (Bazaar) vs. closed source developing model (Cathedral).

12. Debian Community Guidelines, http://people.debian.org/~enrico/dcg/index.html

13. Intro to Distributed Version Control (Illustrated)
    Kalid Azad
    October 15, 2007
    http://betterexplained.com/articles/intro-to-distributed-version-control-illustrated/

14. Ruby Language Site:
    http://www.ruby-lang.org
    Your starting point to the ruby community.

15. P. G. Capek, S. Frank, S. Gerdt, and D. Shields, A history of IBM's open source involvement and strategy -
    http://www.research.ibm.com/journal/abstracts/sj/442/capek.html

16. http://en.wikipedia.org/wiki/Open_source

17. http://www.ibm.com/developerworks/opensource/library/os-license/?S_TACT=105AGX44&S_CMP=LP

18. http://en.wikipedia.org/wiki/Copyright

19. http://en.wikipedia.org/wiki/Intellectual_property

20. http://www.opensource.org

21. http://www.fsf.org/licensing/essays/free-sw.html

22. http://www.internetnews.com/bus-news/article.php/3790721/Cisco+Hit+With+Open+Source+Lawsuit.htm

23. http://www.out-law.com/page-8490

24. GUIDE SME's Deliverable ID: D8.1.1 – Carlo Daffara
http://flossmetrics.org/sections/deliverables/docs/deliverables/WP8/D8.1.1-SMEs_Guide.pdf

25. "From Wikipedia, the free encyclopedia"
    - Software build
      http://en.wikipedia.org/wiki/Software_build
    - Software design
      http://en.wikipedia.org/wiki/Software_design
    - Release management
      http://en.wikipedia.org/wiki/Release_management
    - Software release life cycle
      http://en.wikipedia.org/wiki/Software_release_life_cycle

Books

1. Free Redbook: DB2 Express-C: The Developer Handbook for XML, PHP, C/C++, Java, and .NET

   Whei-Jen Chen, John Chun, Naomi Ngan, Rakesh Ranjan, Manoj K. Sardana, August 2006 - SG24-7301-00
   http://www.redbooks.ibm.com/abstracts/sg247301.html?Open

2. Understanding DB2 – Learning Visually with Examples V9.5

   Raul F. Chong, et all. January 2008
   ISBN-10: 0131580183

3. DB2 9: pureXML overview and fast start by Cynthia M. Saracco, Don Chamberlin, Rav Ahuja June 2006 SG24-7298

   http://www.redbooks.ibm.com/abstracts/sg247298.html?Open

4. The Cathedral & the Bazaar

   Eric S. Raymond
   January 15, 2001
   http://www.amazon.com/Cathedral-Bazaar-Musings-Accidental-Revolutionary/dp/0596001088

5. Software Engineering Approaches for Offshore and Outsourced Development - Vol. 4716

   First International Conference, SEAFOOD 2007, Zurich, Switzerland
   Bertrand Meyer, M. Joseph (Eds.)
   February 5-6, 2007, Revised Papers
   Springer
   Copyright © Springer-Verlag Berlin Heidelberg 2007

6. IBM® Rational® ClearCase®, Ant, and CruiseControl - The Java™ Developer's Guide to Accelerating and Automating the Build Process

   Kevin A. Lee
   First Edition - May 24, 2006
   IBM Press
   Copyright © 2006

7. Clinical engineering handbook - Academic Press Series in Biomedical Engineering

   Joseph F. Dyro
   Chapter 85: Medical Device Software Development
   Richard C. Fries and Andre E. Bloesch
   August 27, 2004
   Elsevier

Imprint: Academic Press
Copyright © 2004 Elsevier Inc.

8. The Art of Software Testing

   Glenford J. Myers
   Second Edition
   Revised and updated by Tom Badgett and Todd M. Thomas with Corey sandler
   John Wiley & Sons, Inc., Hoboken, New Jersey
   Copyright © 2004 Word Association Inc.

9. Producing Open Source Software - How to Run a Successful Free Software Project

   Karl Fogel
   First Edition
   October, 2005
   O'Reilly Media, Inc.
   Copyright © 2005, 2006, 2007 Karl Fogel, under a CreativeCommons Attribution-ShareAlike (3.0) license

10. Innovation Happens Elsewhere

    Ron Goldman & Richard P. Gabriel
    Version 1.01
    http://www.dreamsongs.com/IHE/IHE.html

Contact emails

General DB2 Express-C mailbox: db2x@ca.ibm.com

General DB2 on Campus program mailbox: db2univ@ca.ibm.com

Getting started with open source development couldn't be easier.

Read this book to:

- **Find out what open source development is all about**
- **Learn about open source licenses**
- **Understand how to work with open source communities**
- **Learn about open source business models and trends**
- **Start and contribute to open source projects**
- **Practice using hands-on exercises**

Open source software development is a community-driven methodology to develop software products, from the design and development stages to distribution. Developers across different parts of the world are passionate about their collaboration, and several successful projects including Firefox, Moodle, and Drupal, are widely used today. Moreover, many companies are using open source software as the foundation to build their business models.

This book gets you started into the fascinating world of open source software development. Using the exercises and case studies provided, you will get good hands-on experience to contribute to and start open source projects.

To learn more about open source development visit:
ibm.com/developerworks/opensource/

To learn more or download DB2® Express-C, visit
ibm.com/db2/express

To socialize and watch related videos, visit
channelDB2.com

This book is part of the DB2 on Campus book series, free eBooks for the community. Learn more at db2university.com

9 780986 628337

Price: 24.99USD