A NRCFOSS Series

Enterprise Application development using FOSS

National Resource Centre for Free/Open Source Software (NRCFOSS)
Chennai
(www.nrcfoss.org.in)

List of contributors:

This book is an outcome of the third F/OSS Teachers' Training Programme (TTP-III) conducted by NRCFOSS during 8^th to 16^th June 2006 at the AU-KBC Research Center, MIT Campus, Anna University, Chennai. Much of the material presented here are based on the lecture notes of this TTP, and mainly includes contributions from the following persons :

Mr. Abhas Abhinav	Deeproot Linux Pvt.Ltd, Bangalore
Mr. Bharathi Subramanian	MIDAS Communication Technologies Pvt.Ltd, Chennai
Dr. G. Nagarjuna	HBCSE, TIFR and FSF (India), Mumbai
Dr. M. Sasi Kumar	OSSRC, C-DAC Mumbai
Mr. Senthil Anand	AU-KBC Research Center, Anna University, Chennai.
Prof G. Sivakumar	Dept of CSE, IIT, Mumbai
Prof N.B. Venkateswarlu	Dept of CSE, GVPCE, Visakhapatnam

CONTENTS

Figures and Tables

Chapter 1

F/OSS and Enterprise Applications

Free / Open Source Software (F/OSS) has come up with a fundamentally new paradigm of how software is produced, distributed and used. The basic premise in FOSS is that the source code of the software is released to the user. It is expected that making the source code free and public allows, perhaps even motivates, the users to modify the code and redistribute the derivative software. The question then becomes: at an enterprise level, how does FOSS applications work and how can they be made compatible with the "business" aspects of an enterprise? At the outset, one has to understand the requirements at enterprise level, identify the types of solutions needed and keep in mind the broad nature of the user and developer spectra. The present book attempts to bring these points into focus and highlight some of the open source technologies available for enterprise applications based on a 3-tier architecture.

Chapter 1 introduces the broad framework and ideas that would get expanded and elaborated in the rest of the book. The first two sections of this chapter take up the application examples from Banking and E-Governance sectors, and discuss the enterprise level requirements. Section 3 discusses the system architecture focussing on Layered Architecture Applications that are in vogue. As FOSS solutions are conglomerates of packages from diverse developer groups, it becomes essential to have some standards and norms, which are discussed in Section 4. The business focus on FOSS including resource generation, licences and business models are discussed in the final two sections of this chapter.

1.1 Enterprise Level Requirements

To start with, let us briefly dwell on what an enterprise is and what is the role of software in an enterprise. Any ongoing project or a project to be undertaken can be called an enterprise, especially when it involves some specific scope of work, intricacy, ingenuity and risk. In more common terms, Enterprise refers to an organization involved in an activity with a commercial interest. Based on the size and scope of activities, the enterprises are broadly classified into two groups : Small and Medium Enterprises (SMEs) and Large Enterprises.

SMEs are characterised by low or medium level capital and budget, and higher susceptibility to risks. In particular, small scale industries (SSIs) are often started by first-time technocrats with great resource constraints. Since the preference will be for cost effective technologies and software, SMEs are potentially the prime industry users of open source software. Large enterprises involve high capital investment, with long terms goals and plans. They can afford to take heavy risks and pay for high-cost software.

Enterprise needs software applications that integrate and support all aspects of operations of an enterprise or business, with the following features:

- heterogeneous, geographically distributed, data sources
- diverse user base
- multiple platforms / products / vendors / technologies
- high level of integration and consistency

In general, every enterprise requires the following for their computer and information technology

oriented operations :

- physical infrastructure: hardware, network, humanware.
- System software: OS, middleware, RDBMS, etc.
- Application properties: scalability, reliability, availability, performance, security, trust, privacy, etc.
- systemic properties: integration, interoperability, manageability, etc.

While developing enterprise level applications, one should bear the following factors in mind :

- Application should meet the functional requirements of the business
  -- beware of the changing goal posts and rising expectations!
- Application should be scalable and modular so as to be easily adaptable to expansion of activities, shift in business focus, etc.
- Application should fulfil the functionality aspects such as
  --- meet the expectations of response time (latency) and network availability
  --- minimal error rate, maximal number of transactions
  --- allowable number of repeat failures per link / location / component
- Installation and Support Services
  --- installation time guarantees, terms and conditions
  --- vendor problems and escalation policies
  --- disaster back-up commitments
  --- help desks to cater to different skill levels of users
  --- considerations related to system uptime, mean time to repair
  --- clear description of services provided
  --- continuation and extension of services
  --- provisions related to break or termination of contracts for non-performance
  --- penalties for failures to meet service levels
- Application Security, a process and not merely a product, which includes
  --- user authentication systems indicating privileges and restrictions
  --- encryption of data
  --- intrusion detection and intrusion handling software
  --- firewalls and other security related measures
- Measurement systems for determining and monitoring service levels

In particular, user administration and providing support services might become a nightmare, unless they are well planned based on policy guidelines finalised with the user group. The Quality of Service (QoS) parameters descriptions and guarantees should be well laid out and covered in the Service Level Agreement (SLA) between the developer group and the user enterprise.

1.1.1 Banking Applications – An example of cheque truncation system:

Banking and financial domain is one of the major customers of open source softwares. The reason why financial institutions are moving to Open Source are freedom of choice, being able to adapt and change the software, support for standards, flexibility, better quality through the open development process and the possibility of community involvement - to name a few.

Despite official edicts to the contrary, there is an enormous amount of open source work going on in the banking domain. One of the reasons why open source software is best for banking is that they have fewer security flaws than proprietary software. With more eyes able to look at the underlying source code, bugs should be found and squashed much faster.

Let us consider the example of a software solution related to ***Cheque Truncation System (CTS)***. Cheque truncation is settlement of clearing transactions on the basis of images and electronic data without the physical movement of the instruments. The Clearing Cheque is truncated at the

Presenting Bank itself. Normally, the instrument is physically sent to Payee Bank, which involves time delay, especially in a huge country like India, and also possible damage or loss due to the huge number of such instruments in transit. CTS allows a financial institution to truncate cheques at the "Point of Capture" by enabling the presentation of cheques to the "Paying Bank" electronically and process Return cheques electronically.

In India, the Reserve Bank of India had given the following guidelines for clearance of cheques :

- truncation within the presenting bank
- point of truncation to be decided by presenting banks
- settlement on current MICR data
- electronic images for payment processing
- grey scale technology
- inter-operable and open standards
- image preservation for eight years
- Public Key Infrastructure Security for images and data flow

The proposed solution from a committee set up by State Bank of India had the following features:

- opt for Branch Model (decentralised scanning) for outward clearing from all SBI Group branches
- Desktop Scanners of varying speeds to be installed at branches depending on volumes received
- Central Pool of scanners to provide for redundancy in case of breakdowns or increased volumes
- Data to travel to Service Branch through SBIConnect
- Actual outward clearing presentment done centrally from Service Branch
- Inward Clearing to involve handling only electronic records using software at Service Branch
- Backward integration with Core Banking / Bankmaster system for posting/return cheques, etc.

The proposed solution set is given in Fig. 1.1.

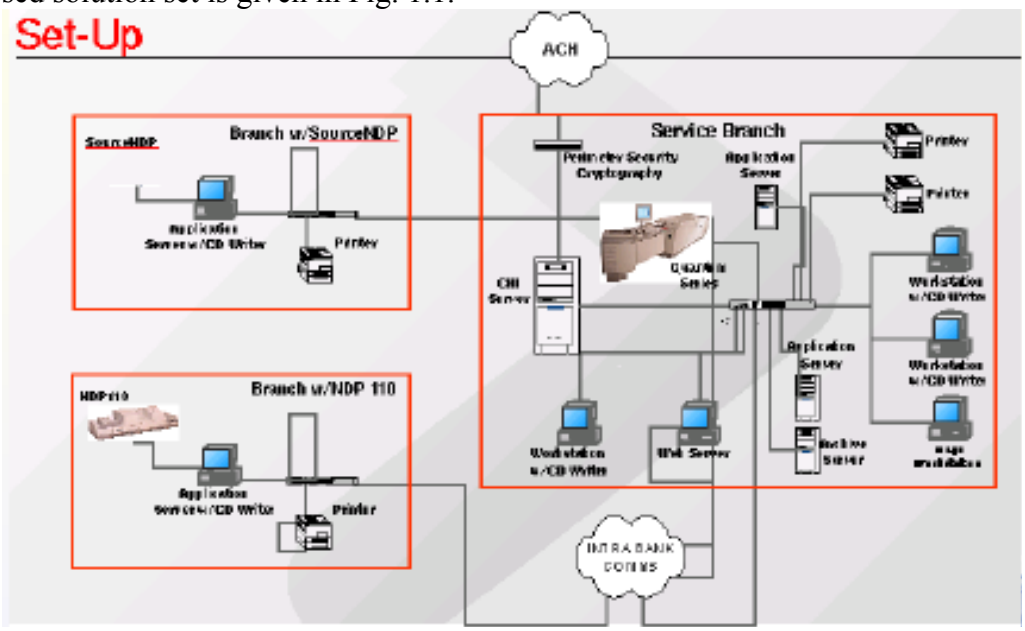

Fig. 1.1 Set up Cheque Truncation System

A modular architecture for the CTS solution is given in Fig. 1.2.

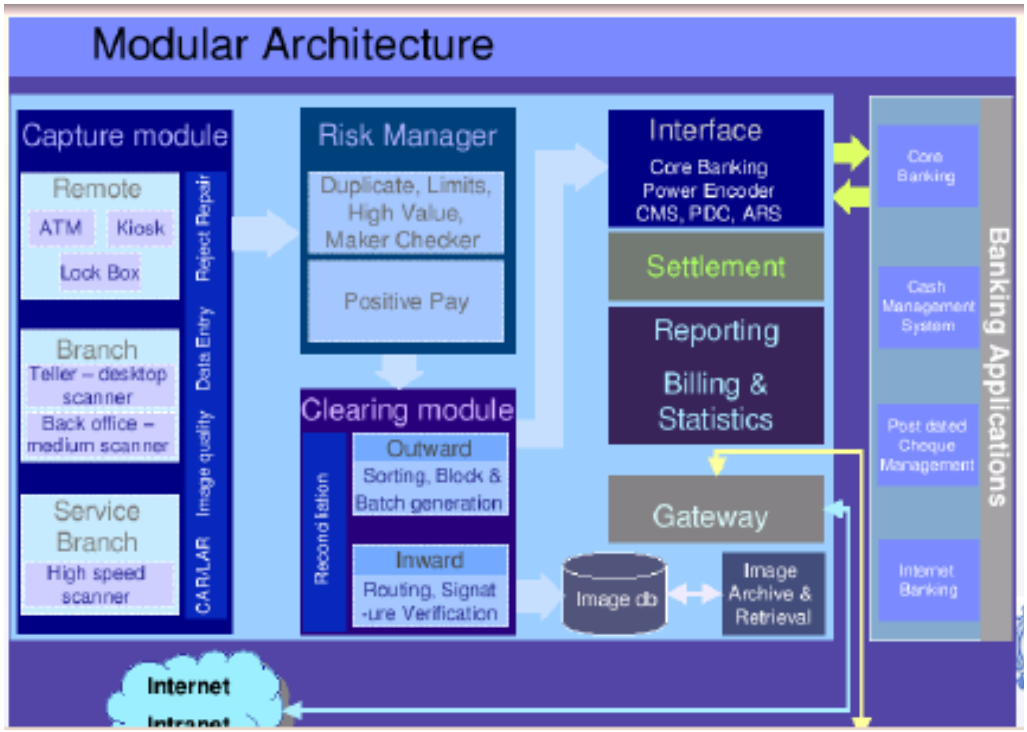

Fig. 1.2 Components of solution for Cheque Truncation System

1.1.2 Open-source software in e-Governance

The aim of e-Govenance is to provide efficient, convenient and transparent services to create easier and smoother interaction between the public sector and the citizens. In practical terms, e-Governance involves : Use by public administrations of technology, primarily internet-based technology, which contributes to improving access to and supply of public information and public services to citizens, companies and the public sector as a whole. The need for wider deployment of public utility services, the focus on access to information, etc., demands huge investments in Information Technology. In India, for instance, e-Governance will cover a wide variety of domains (Fig. 1.3) like education, civil supplies, taxation, etc., in the various states and union territories, involving huge resource requirements

. ***Fig. 1.3 e-Governance model***

It is therefore natural that a comprehensive assessment is made in connection with these investments – as to the technology to be used in development and implementation, the intended application domains, and control of the development process and ownership of the underlying technologies used in e-Governance. In particular, the application usage and future enhancements should not be hampered by vendor lock-ins, forced updates, and other inhibiting factors. This increases interest in the opportunities opened up by open-source software, and makes the question of the potential for the use of open-source software in e-Governance economically advantageous and relevant. Open-source software also has a great impact on the political agenda, nationally and internationally, as it provides freedom from monopolies.

The possible advantages of Open-source software over proprietary software include the following :

- Possibility of savings and economic advantages : open source is not necessarily free, but in most cases is considerably cheaper than proprietary software.
- Open standards and possibility of integration: the principle for the development of software is established in public forums, in contrast to proprietary standards, which are kept secret.
- Security and Quality: Open source code enables the user and other interested parties to check whether the program is written in a justifiable way and wherever applicable, to identify elements of danger to the stability and security of the code.
- Supplier independence and avoidance of vendor 'lock-in'
- Possibility of customisation, re-use and enhancement
- support for entrepreneurship and business formation, especially for firms that would otherwise be unable to withstand corporate competition
- local IT industry and digital self-sufficiency, local content creation and consumption

.

1.2 Some FOSS products for Enterprise applications

Software sharing has been around as long as software itself. In the early days, the competition was focussed on the hardware sector and software was rarely considered as a business asset. Modifications and extensions for software, according to the manufacturers, made the hardware that much more attractive to potential customers. However, the sharing was constrained by the physical process of copying and distributing the modified codes. With the advent of internet, it became easier to enable widespread, frictionless sharing, as is currently in vogue. Since this could cut into business, manufacturers embarked on stricter enforcement of copyrights and the "proprietary regime" allowed only distribution of executables rather than the source code. It was an irony that just as internet was making unobstructed software sharing technically possible, the companies, from a presumably business point of view, denied access to code that ran the users' machines or insisted on strictly enforceable non-disclosure agreements.

Free/Open Source Software (F/OSS) challenges our preconceptions about how software is used, produced and distributed. FOSS refers to software that has made its source code free and public, and allows – perhaps implores – users to modify the source code for customised and/or enhance features, and redistribute the derivative software. While liberating the source code is a goal in itself, FOSS encourages, or even obliges, programmers to give other programmers and users those same freedoms and opportunities. To be precise, the emphasis is on Free as in Freedom, thereby allowing that a user could distribute the software at a cost.

FOSS products and packages are now becoming available to meet almost all application needs, be system software, programming tools, utilities or generic applications.

Systems: Operating systems written in FOSS have mushroomed in the last decade. Apart from commercially important ones such as *Linux, BSD* and *Open Solaris*, there are a large number of open implementations of OS variants like *Be*, *Darwin*, and *QNX RTP*. Networks are currently synonymous with systems, and the core *BSD* stack for TCP/IP has been heavily optimized for performance. A large number of stacks for other network protocols, including 802.xx, are available in FOSS.

Programming Tools: There are a large number of tools available in open source. Compilers exist for almost every programming language, including procedural languages such as *C* and *Java*; extended scripting languages such as *Perl* and *Python*; inferential languages such as *Prolog* and functional languages such as *LISP*. Compilers for languages such as *Lex* and *Yacc* are also available. Beginning with ***Emacs***, there are now a large number of text editors, many of them extensible. Since 2000, there have a number of Integrated Development Environments (IDEs) available in FOSS, *Eclipse* being a well known example.

Utilities: *Gimp* is a good stand-in for Photoshop-like capabilities. *Evolution* and *Thunderbird* are modern-day user-interface based clients. Open Instance-Messaging clients like *Pidgin* (formerly *Gaim*) are also available. The *Mozilla* effort has evolved into the current day *Firefox*.

Office tools: Wherever computer is used, a suite of office tools that includes applications such as, word processor, presentation, spread sheet, database etc. is a basic requirement. The open source community has developed several tools including *Open Office*, *Abi Word*, *KOffice* and so on. Some of these tools, *Open Office* for example, provides compatibility with proprietary software that enables the users to continue using the existing documents.

e-Commerce tools: The e-Commerce is one of the most important developments after the proliferation of the web. Several tools are required to set up an e-Commerce portal. Not all companies that want to sell their products through the web can afford

to either develop their own tools or spend huge money on proprietary software. In this respect, *Zen Cart*, os*Commerce* and *XT-Commerce* are some of the e-Commerce software prevalent. These tools have modules for cart management and content management, including the database and query functionalities for managing the content data.

Open source has again proved its worth by making available several applications including Document Management Systems (DMS) - *Knowledgetree, Alfresco, DocMgr* etc. ; Database - PostgreSQL, MySQL ; Knowledge Management systems – *OpenCyc, Owl,*l etc. Some common commercial software and their equivalent FOSS options are given in Table 1.1.

Application	**Commercial software**	**FOSS Option**
Operating System	*Windows, OS X, Solaris*	GNU/Linux(Debian, Redhat, Fedora, Ubuntu, Mandrake, SuSE, etc.) Free/Open/Net BSD, Minix, Open Solaris, etc.
User-Interface Manager	*Windows UI, Finder, OpenView*	*KDE* (K Desktop Environment), *Gnome*, *IceWM*, *Window Maker*
Networking	Windows NT, Cisco IOS software,	NAT (Router), iptables (Firewall), Snort (IDS), Nmap (port mapper), Ethereal, Dsniff (sniffer tools), OpenSSH, OpenSSL (Communication Tunnel), SAMBA (File Server), CUPS (Print Server)
Distributed Computing	Cray	Beowulf , Mosix, PVM, MPI, MOSIX, NAMD
Database Tools	Oracle, Microsoft SQL	MySQL, PostgreSQL, Firebird, druid, GNUdata
Web Applications	IIS, Personal Webserver	Apache, Tomcat (Webserver), Sendmail (mail server), BIND (DNS Server), Squid (Proxy server).
Enterprise Applications		EZPublish, Smart, PostNuke?.
UML	Rational Rose	Eclipse
Browser Tools	IE, Safari, Opera	Mozilla, Firefox, Galeon, Netscape, Konqueror.
Desktop Applications		Jabber, Mplayer, Xine (Multimedia), GIMP, Dia (Graphics), LinuxCAD, VariCAD (CAD), Acrobat, Xpdf, Ghostview (PDF/PS)
Scientific Applications	Matlab, Maple, Mathematica	SciLab, Octave, GNU Plot, Singular, ORSA, Potnia, Maxima, Axiom, Pari, GPS, LAPACK, LINPACK, Gstat
Bioinformatics & Computational Biology		BLAST, EMBOSS, COG, InterPro?, Genesis, E-CELL, Rasmol, Argus Labs, ImageJ, System Biology workbench, VMD, XCOSM
Office Applications	Microsoft office suite, WordPerfect? Suite, ixx	Open Office, Koffice, Abi Word, Mr Project, GNU Cash, PS Slides. Presenter, Magic point, Gnumeric, Gnuledger, Oleo, Qhacc, Sharp tools spreadsheet
Mail clients	Microsoft outlook	GNUmail, Kmail, Pygmy
Personal information manager		Boob, evolution
Address book	Windows address book	kalarm, Multiuser address management system/book, rubrica
Calendar	iCal	Acal, active calendar, pcal
Project manager		Gforge, Savane, cuteflow

Web Development		PHP, Perl, Netbeans, Aspire, Jflash, Free Pascal.
Security		OpenCA, Open VPN, Open L2TP, VPN Dialer (IPSec), Axcrypt, Network Security Toolkit (NST), Snort, OpenNMS, nessus (in OSSIM), INSERT, Trinux, jGuard, NSAT.
Content Management System	Vignette, Interwoven	Plone, CMS, Nucleus CMS, A Tutor, PHPNuke
Publishing		Latex, MathML, XML, OpenJade?, Scribus, SodiPodi?, Inkscape. Doxygen,TexMacs?
Software development	Microsoft visual studio, Xcode	GCC, Java, FORTRAN, Perl, Python, Ruby, Lisp, Ocaml, Glade, Qt
Wireless		ZoneCD, Wellenreiter, AirSnort?, AirJack?, Tiny OS (sensor n/w's), Prism Stumbler, AirTraf?.
Electronic Design Automation (EDA)		GEDA, tkgate, Gaphor, Pcb, PIPE (petrinets), Xcircuit, GTKWave, EngyCAD, Icarus Verilog, ViPEC, Alliance, Darwin2K, HDLMaker.
Project Tracking / Management		Achievo, Gforge, Savannah, CVS, Subversion, Bugzilla, Gnats
Market Research / Data Analysis		CCOUNT
Data Visualisation		Flounder, Gri, ImLib3D, Vis5D.
E-Commerce		DotGNU, Goldwater, Bidwatcher, TCLink, Interchange, etc.
Geography	Intergraph	GRASS, GDAL, DGPS, GpsDrive?, GPStk, Height Map Generator, RoadMap?, Thuban.
Entertainment	Windows mediaplayer, winamp.	Xine, xmms, fftv, Kplayer
Graphics	Photoshop, 3D studio max	Gimp, Artofillusion, Ayam, Blender

Table 1.1. Sample set of proprietary software and FOSS equivalents for various applications

1.3 Layered Architecture Applications

Whether it is open source or proprietary, the most important element of the application design is the system architecture which defines how the various parts of the application interact with each other. There are basically 3 types of architectures and can be classified based on the number of layers or parts between the number of users and data. : Single-tier Architecture, 2-tier Architecture, 3-layer Architecture.

Single-tier Architecture:

All services are either located on a single host sufficiently provisioned with memory and CPUs, or deployed on multiple servers, with each server hosting all of a particular component product's services. In single-tier end users communicate directly to the data stores and not through proxies or other agents. Examples: Main Frame computer with dumb terminals, CMC's original railway reservation system. The main disadvantages are :

- lack of "network" constrains the functionality
- the interface available to the public or a typical client is not very "user-friendly"
- the interactive features may not cover the full requirements, e.g. Payment options in finance-

related applications

2-tier Architecture :
The access and data layers for the component products are separated onto different servers. In two-tier client/server architectures, the client interface is usually located in the user's desktop environment and the database management services are in a more powerful server that services many clients. The user system interface environment and the database management server environment split the processing management duties. The database management server contains stored procedures and triggers, to handle and store data ensuring integrity of the contents. The server sends out data based on SQL queries submitted by the application. The application on the client computer receives the data and presents it in a readable format to the user. The networking environment allows for distribution of data storage and computing tasks as against the centralized solution in the single tier architecture. The client is not a dumb terminal, but should have some computing capability to assimilate and present the data.

Many applications / protocols in Internet including e-mail, FTP, and web applications use this architecture. As an example, let us suppose there is a piece of software that students at a school can use to find out their current grades. The structure of the program is such that the database of grades resides on the server, and the application resides on the client, e.g., the desktop the student is physically interacting with. The student runs the client-side program (by clicking buttons, menu options, etc). The program fires off a query to the database, and the database responds with the student's grades. Now this application uses all this data to calculate the student's grade, and displays it for him.

3-Tier Architecture:
Two-tier architecture is fine for a small number of users with similar profiles, often running the same operating system. In reality, however, the client needs are diverse and their systems could vary from low-end PCs to powerful workstations. An enterprise may have many branches with employees interacting with systems at varied levels. The points one should consider include the following :

- Variety of users/clients with different levels of computer skill
- Geographically distributed locations of users and systems
- Variety of client side systems including dumb terminals, workstations, hand-held devices
- Support for different operating systems, especially at the client side
- Varied levels of access, e.g., network connected, dial-up, cellular, etc.
- Expectations for response time and network availability
- Anticipated load of transactions
- Need for different levels of help
- Security Risks.

The 3-Tier architecture considers these aspects and introduces an intermediate layer to enable the server to handle different types of clients. The Presentation Layer deals with the client aspects while the Data Store Layer is concerned about the server side. The inter-connectivity is provided by an applicaiton service segment which does the necessary translations of queries and data to ensure smooth client – server interactions. The basic structure is as in Fig. 1.4, given below.

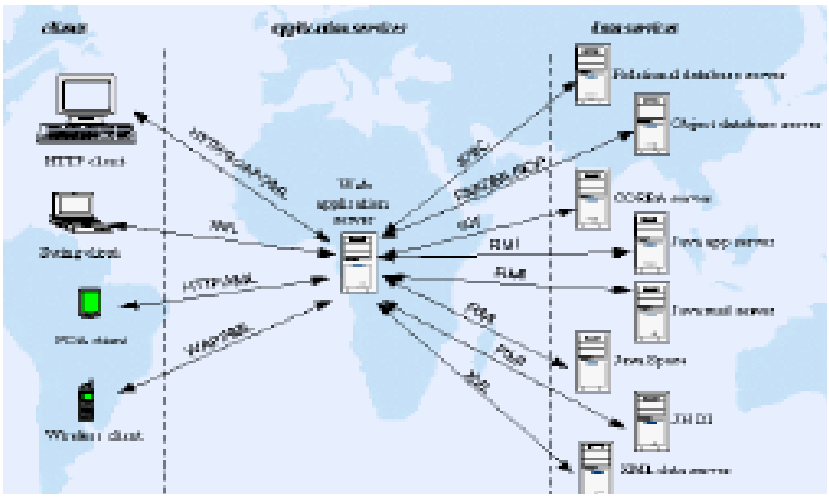

Fig 1.4 3-tier Architecture

In this book, the 3-tier architecture under Model-View-Control paradigm is considered for providing solutions for enterprise applications. The three layers -- Presentation, Data Store and Business Logic – and the associated technologies are discussed in detail in Chapters 3 onwards.

1.4 Open Standards

Open IT standards are even more important in this present information age of ubiquitous internet and the covergence of information technology and communications. No single technology, group or vendor can provide for everything and, therefore, interoperability in a heterogeneous environment is required more than ever. It is only by strict adherence to standards and specifications that a high degree of interoperability can be achieved. Standards that are open and non-discriminatory are preferred because there is no dependence on any single entity, all types of products can implement them and all interested parties can partake in their development.

The dictionary meaning of *standard* is a) a level of quality or attainment, and b) something used as a measure, norm or model in comparative evaluations. In technical parlance, standard is "a specification that has been approved by a recognised organisation or is generally accepted and widely used throughout by the industry" . Open Standards implies that the specifications are publicly available for achieving specific tasks. Some of the official definitions are :

- ***The Open Standards Policy of the State of Massachusetts, USA*** defines it as specifications for systems that are publicly available and are developed by an open community and affirmed by a standards body.
- The European Commission's ***European Interoperability Framework (EIF)*** defines open standards as something that is available either for free or at a nominal charge for usage, copying and distribution and any patents present are to be made irrevocably available on a royalty-free basis, and there should be no constraints on the re-use of the standard.

According to the renowned Open Source exponent Bruce Perens, an open standard is more than just a specification, and that the principles underlying the standard and the practice of offering and operating the standard are what make the standard open. The Perens definition has found wide acceptance among the FOSS communities worldwide. The Perens principles for open standards are

:

- **Availability** for all to read and implement.
- **Maximize end-user choice** by creating a fair, competitive market for implementation and avoid vendor lock-in
- **No royalty** so that they are free for all to implement. Certification of compliance by the standards organization may involve a fee.
- **No discrimination** based on implementer or organization for any reason other than the technical standards compliance of a vendor's implementation. Certification organizations must provide a path for low- and zero-cost implementations to be validated, but may also provide enhanced certification services.
- **Extension or subset** of implementations subject to meeting compliance criterions from certification organizations
- **Predatory practices** to protect against subversion of the standard by embrace-and-extend tactics. The licenses attached to the standard may require the publication of reference information for extensions, and a license for all others to create, distribute and sell software that is compatible with the extensions. An open standard may not otherwise prohibit extensions.

A standard is 'open' when :

- it cannot be controlled by any single person or entity with any vested interests
- it evolved and is managed in a transparent process open to all parties
- it is platform independent, vendor neutral and usable for multiple implementations
- open published
- available royalty-free or at minimal cost, with any other restrictions offered on reasonable and non-discriminatory terms
- approved through due process by rough consensus among participants

Some of the examples of Open Standards are:

1. The POSIX (Portable Operating System Interface for UNIX) from the Open Group, IEEE and ISO
2. The Tranmission Control Protocol/Internet Protocol(TCP/IP) suite of networking protocols from the Internet Engineering Task Force (IETF)
3. The Hypertext Transfer Protocol (HTTP) service protocol from the World Wide Web Consortium (W3C) and the International Organization for Standardization (ISO)
4. The Unicode coding standard from the Unicode Consortium and ISO

Many people are confused between the terms open standards and FOSS thinking that they are one and the same or one cannot exist without the other. To be consistent with other publications by IOSN, the term FOSS will be used in this document to refer to open-source software and/or free software. Unless otherwise stated, open standards is not the same as FOSS, which refers to software that follows certain principles in its creation, modification, usage, licensing and distribution. In particular, FOSS should have the four fundamental freedoms: 1. freedom to run the program, for any purpose; 2. freedom to study how the program works, and adapt it to your needs; 3. freedom to redistribute copies so you can help others; and 4. freedom to improve the program, and release your improvements to the public

FOSS is software whereas open standards refer to standards. Widespread usage of standards, and especially open standards, is very important to FOSS. It makes it easier for FOSS to be compatible with proprietary software. It is a practical reality that FOSS needs to coexist with proprietary software and that compatibility with the proprietary platforms is facilitated if standards are adhered to by all. If all software products were to follow standards strictly, they should be able to interoperate and communicate among themselves well and data files could be read and written transparently. While both proprietary and open standards may allow this to happen, the latter are preferred by the FOSS community as they facilitate free access, open development and participation.

FOSS is Useful for Popularizing Open Standards

FOSS can play a useful role in popularizing an open standard. A FOSS implementation of a standard usually results in an open and free-working reference implementation. A lot of the benefits of open standards are negated if its only implementation is a closed and proprietary one. The availability of a FOSS implementation will spur quicker adoption and acceptance of the standard as everyone has easy access to the implementation of the standard and so can try and test it out. A very good example of this is the Internet HTTP standard. One reason why this service became universally accepted is that very early on, there were free and open implementations of both the HTTP server (e.g., National Center for Supercomputing Applications or NCSA HTTPd, Apache) and client (e.g., NCSA Mosaic).

Benefits of using open standards

Numerous benefits are obtained if an organization ensures that its technological and IT procurements and implementations follow open standards as far as possible. First and foremost, there is less chance of being locked in by a specific technology and/or vendor. Since the specifications are known and open, it is always possible to get another party to implement the same solution adhering to the standards being followed. Another major benefit is that it will be easier for systems from different parties or using different technologies to interoperate and communicate with one another. As a result, there will be improved data interchange and exchange. It will not be necessary to use the same software or software from a particular vendor to read or write data files.

If open standards are followed, applications are easier to port from one platform to another since the technical implementation follows known guidelines and rules, and the interfaces, both internally and externally, are known. In addition to this, the skills learned from one platform or application can be utilized with possibly less re-training needed. This can be contrasted with the usage in applications of proprietary standards that are not openly published and where there is inadequate information publicly available about them

National Considerations

From the national viewpoint, the usage of open standards by a government is even more important. In this information age, a government will need to use IT solutions to ensure that it has adequate and reliable information to enable it to govern the country effectively. It is vital that these IT implementations make use of standards that are open as far as possible. In cases where open standards are not available, the government may want to consider other standards that are freely available for usage and implementation. It should also take into consideration how open these standards are and whether they have the possibility of becoming open standards later.

All these can help ensure that there is less likelihood of its information systems being locked in later by any single technology or product. It is also in the interests of national security that open

standards are followed to guard against the possibility of over-reliance on foreign technologies/products. Imagine the implications to a sovereign nation if the electronic records of its citizens are kept in databases that can be accessed readily only by proprietary software from a foreign vendor or the documents of the government are kept in a format that belongs to a vendor who thus has total control over its accessibility both now and in the future.

e-Government Projects Specify Open Standards

Many countries have started on e-government projects or initiatives, most of which have policies stating that, as far as possible, open IT standards and specifications are to be followed. Countries that have such policies include Norway, Denmark, the United Kingdom, the Netherlands, France, Brazil, Australia, New Zealand, and Malaysia.

The European Union's EIF, a framework to facilitate the interoperability of its member countries' e-government services, recommends the use of open standards for maximum interoperability.

In addition, more and more public sector agencies all over the world have adopted or are considering adoption of policies that require open standards.

Particular Benefits of Open Standards

File Formats Modern information systems generate data (lots of it in many cases) that have to be stored in some form of electronic file formats for efficient storage, retrieval and exchange. If their specifications are not publicly known, only software and systems from the owner of these proprietary formats can readily access them. In some cases, while the format may be known, it may be the property of a particular party and this party may control the way the format evolves or is used. In such cases, users can have very little say or control over the format. Also it may be possible that the owner may not publish the format specifications at a later stage for a new version. So while compatible systems can be created that can access the files now, there is no guarantee of this when a newer version comes out.

Internet Services and Applications The Internet is perhaps the best showcase of how when technologies are implemented using mainly open standards, there is almost universal accessibility, acceptance and benefits. Most networking infrastructure of the Internet is implemented based on open standards drawn up by IETF. As a result, today, it is possible for one to access major services offered on the Internet using a multitude of environments ranging from commodity PCs, hand-held Personal Digital Assistants (PDAs) and mobile devices to proprietary set-top black boxes and TV sets. Without this adherence to open standards, the Internet would not be as ubiquitous as it is today.

Open Standards Organizations are usually non-profit and may be government-appointed, industry-backed, non-government organizations or even voluntary ones. Some active organizations that are generally perceived to be open include IETF, IEEE, OASIS, W3C and the Free Standards Group (FSG).

1.5 FOSS Licenses

A software license consists of a set of rights, permissions, and restrictions related to purchase and use of software. Under a software license, the licensee is permitted to use the licensed software in compliance with the specific terms of the license. If there is a breach of the license, depending on the license it may result in termination of the license, and potentially the right of the owner to sue.

1.5.1 Copy right

Copyright is a set of exclusive rights regulating the use of a particular expression of an idea or information. At its most general, it is literally "the right to copy" an original creation. In most cases, these rights are of limited duration. The symbol for copyright is ©, and in some jurisdictions may alternatively be written as either (c) or (C).

Compared to other legal concepts, copyright is a relatively new invention in human history. The development of copyright regulation reflects the social and technological transformation around human creative activity and distribution of the resultant profits. While granting exclusive private rights to authors or copyright holders has been considered as a way of encouraging human creative activity, copyright law also claims to recognize the larger public interest, particularly with respect to education, research and access to information.

Copyright applies to the expression of ideas in different forms, including literary, dramatic, musical, artistic, and other intellectual works. The ideas expressed in such works are themselves not copyrightable. Since the 1980s, the copyrightability of software became internationally accepted and it has become an international trend for copyright to be applied to computer software. The WIPO Copyright Treaty (1996) also states that computer software should be regulated by copyright law.

1.5.2 Royalties

Sometimes simply referred to as "royalty", it refers typically to the sum of money paid to the proprietor or licensor of intellectual property (IP) rights for the benefits derived, or sought to be derived, by the user (the licensee) through the exercise of such rights. Royalties may be paid for the use of copy right, patent, trademark, industrial design, procedural knowledge or a combination of them. However, the inherent nature of Free/Open Source Software contradicts the notion of royalty. One could term the fee payable under some of the Open Source licenses, discussed in the following sections, as the 'royalty'.

1.5.3 GNU General Public License (GNU GPL or GPL)

In 1985, the Free Software Foundation (FSF) was established to promote the idea of free software, strongly avowing the rights of non-copyright holders. To ensure that these freedoms are not mere slogans, the **GNU project** was launched in 1984 to develop and complete a UNIX-style operating system which is free software, i.e., the GNU system, and the project went further to include other application software.

Under the existing legal structure, once the work is created, copyright protection is granted to the exclusive disposal of the copyright-holder. Without an explicit expression, it is assumed that you, the copyright-holder, claims all the rights granted to you, and that any dissimilar subject opinion must be ignored. That is to say, the law burdens you with an explicit expression not to claim some or all rights granted to you.

While people might sometimes not know how to make such an explicit expression, the creation of the GNU General Public License (GNU GPL or GPL) serves as a legal tool to bring out the freedoms stated in the Free Software Definition and to maintain the environment that supports free software

The GPL is a license, but it is different from proprietary licenses. It grants the users the rights that were considered the exclusive right of the copyright-holder by law and by business practices, including the right to access to the source code, the right to run the program, the right to make

copies and redistribute the copies, and the right to modify the program and distribute the modified program. On the other hand, although the GPL grants the users many rights and freedoms to use the software, it also sets certain limitations in order to ensure that free software, and its derivations, remain as free as it is.

When a work is licensed under GPL, it means that you, the author of the software, still claims copyright of the work, but can adopt a different license as a way of explicit expression to allow the public to have greater freedom to use your work than copyright law assumes.

The GNU General Public License (GNU GPL or GPL) is the classic free software license. It is also the most well-known and widely adopted among all FOSS licenses. The GPL is an invention brought out to fulfil the freedoms defined by free software. It is both a license and a document to manifest the basic idea behind free software.

Copyleft is the way in which GPL guarantees freedom. Copyleft prevents free software from being turned into proprietary software again. It uses copyright law, but serves the opposite of its usual purpose. Instead of becoming a means of privatizing software, copyleft uses rights granted to authors to keep the software free.

Instead of being a work in the public domain that everyone is free to make use of, a GPL-ed work or a copyleft-ed work is still a copyrighted work. The author does not give up rights as a copyright-holder, but the rights are exercised in a way different from traditional proprietors. The software is not released in the public domain exposing it to the danger of being privatized again. Instead, the claim to the rights is retained, and used to regulate the ways in which people make use of your work. By licensing the work under the GPL, one allows the users to have the rights allowed by the free software movement, and also ask the users to take on some responsibilities to keep the software and its derivative works as free as the way you intended your work to be.

Major terms and conditions of the GPL include the following :

- User can run the program, make copies of it and redistribute, even for commercial purposes, provided an appropriate copyright notice and disclaimer of warranty are retained ; redistribution in the object code or executable form is also possible, so long as source code is available for all recipients ; can prepare and distribute derivative works of the program, provided the derivative works are also licensed to all third parties under GPL
- No warranty: Although the distribution of the work may be commercial, the work itself is licensed free-of-charge. The distributor can choose to offer warranty protection in exchange for a fee.
- License issued directly from the author: The work is not sub-licensable. When a program is redistributed, the recipients still receive the license from the original licensor, and the redistributors may not impose any further restrictions on the recipients' exercise of the rights granted in the GPL
- Acceptance and termination: By modifying or distributing the GPL-ed program, a person indicates his or her acceptance of the license. The license grant is irrevocable, but when the licensee violates the license, the rights granted will be terminated automatically..
- Co-existence with other legal obligations? The GPL does not concede to any contradictory conditions that are imposed on the recipients. Compliance with the license, if it is not possible as a consequence of a court judgment, an allegation of patent infringement, or for any other reason, the recipient may not redistribute the program at all A GPL-ed program cannot be incorporated into a proprietary program, nor can it be linked with a proprietary

library.

The full text of the GPL is available at http://www.gnu.org/licenses/gpl.html The FSF also maintains a FAQs section about the GPL, which can be accessed at http://www.fsf.org/licensing/licenses/gpl-faq.html

1.5.4 GNU Lesser General Public License (GNU LGPL or LGPL)

Aside from the GPL, the GNU project offers a special kind of copyleft license for libraries. The GNU Lesser General Public License (LGPL) permits LGPL-ed libraries to be linked with proprietary software. Such an exception may be found in different situations. It might be a strategic decision in order to encourage proprietary applications on the GNU system. And for a free library, whose features can largely be replaced by other proprietary features, release under the LGPL rather than the GPL encourages its wider use, which helps to make more improvements in it, and, with a larger body of free software users, garners wider support for free software.

Free software advocates, however, still encourage people to use the GPL for their libraries rather than the LGPL, especially for those libraries which possess significantly unique capabilities. People who are interested in utilizing such GPL-ed libraries will have to release their modules as free software as well, resulting in more useful modules and program being developed within the free software environment.

Major terms and conditions of the LGPL

The LGPL is identical to the GPL in many ways, including the clause that disclaims warranty, which states that the license is issued directly from the author, when the license is to be applied and terminated, and its relationship to other legal obligations applied upon the users. But when it comes to users' rights, the LGPL distinguishes between two different kinds of situations. When one uses a library, a "work based on the Library" means either the Library itself or any derivative work under copyright law , while a "work that uses the Library" means a program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it.

Works based on the Library

In the case of a "work that uses the Library", i.e. the Library itself and its derivative works, the terms are very similar to those outlined in the ordinary GPL. User's Freedoms include running, copying and redistributing, even for commercial purposes, provided an appropriate copyright notice and disclaimer of warranty are retained. Redistribution in the object code or executable form is also possible, so long as source code is available for all recipients. One could prepare and distribute derivative works of the program, provided the derivative works are also licensed to all third parties under LGPL. In addition, one could opt to apply the terms of the GPL instead of the LGPL to a given copy of the LGPL-ed library, especially while incorporating part of the code into a program that is not a library.

Works that use the Library

In the case of a "work that uses the Library", the work itself is not subject to the LGPL. But when linking the "work that uses the Library" with the Library, an executable which is a derivative work of the Library will be created, and such an executable is covered by the LGPL.

Such executables, although the LGPL allows the author to distribute the object code and license it under the terms of his or her choice. It also requires that those terms permit modification of the work for the customer's own use and reverse engineering. When distributing the executable in its object code, the choice is to either distribute the Library together, provided the source code of the Library is made available in ways similar to the distribution of GPL-ed programs; or to not distribute the Library together, but only use a suitable, shared library mechanism to link with the

Library. By creating such a category, the LGPL opens a way for LGPL-ed libraries to be used in proprietary programs. The full text of the LGPL is available at :
http://www.gnu.org/licenses/lgpl.html

1.5.5 Berkeley Software Distribution (BSD)-style Licenses

The Berkeley Software Distribution (BSD) License was first used for the Berkeley Software Distribution, a version of UNIX developed at the University of California at Berkeley. It is easy to follow the BSD license template to create your own license by changing the values of the owner, organization and year appearing in the copyright notice and in the license. Unlike the copyleft licenses, the BSD-style licenses are relatively simple, and have only limited restrictions regarding the software's use.

*Users' Freedom:*1. Make copies of the program and redistribute the program, either its source code or its binary code. The distributor is not obliged to provide the source code. 2. Prepare derivative works and distribute the derivative works, either their source code or binary code. You, the author, are free to choose either FOSS or proprietary licenses for the derivative works. 3. Incorporate the program into proprietary software.

1.5.6 Multiple Licensing

It is important to note that a work can be licensed under more than one license. A license is a choice by you, the copyright-holder, regarding what you think of the relationship between your work and its possible users. But there might be more than one kind of user, and more than one possible relationship. You are, therefore, entitled to choose different kinds of licenses to accommodate different situations.

Take OpenOffice as an example. OpenOffice is dual-licensed under the GPL and the Sun Industrial Standards Source License (SISSL). Although OpenOffice clearly states that it encourages its users to use the GPL in order to participate fully in the OpenOffice.org community, the SISSL is provided as an alternative for those developers and companies who are unable to use the GPL.

1.5.7 GNU Free Documentation License (GNU FDL or FDL)

Good documentation and manuals are extremely important for FOSS, but when they are not licensed as free/open in the same way as FOSS programs are, it is hard for people to make good use of the FOSS programs which they are supposed to illustrate. However, the software licenses do not cover documentation appropriately. The following licenses for software documentation are in vogue.

Although mainly used as a license for software, the GNU GPL itself can also be used on work which is not software, as long as it is clearly defined what the "source code" means when adopting the license. The FSF also provides a license especially designed for documentation. The GNU Free Documentation License (GNU FDL or FDL) is a form of copyleft license for use on a manual, textbook or other document to assure users about the effective freedom to copy and redistribute it, with or without modifications, either commercially or non-commercially.

By applying the GNU FDL to a document, similar to applying the GPL to a program, you, the author, are granting users to right to make verbatim copies of your work. Since it is a copyleft license, it requires that the copying and distribution of the modification of the FDL-ed work also be licensed under the FDL. But unlike the GPL for software, the FDL has special requirements for making large quantities of verbatim copies.

1.5.8 Creative Commons Licenses

Inspired by FOSS development, the Creative Commons Cooperation advocates the openness of digital content. It urges a more reasonable and flexible layer of copyright in the face of increasingly restrictive default rules.

In 2002, the first version of the Creative Commons Public Licenses (CC licenses) was released. Eleven different licenses identified major concerns which authors consider important. These include:

1. Attribution (BY): For any reuse and distribution, it is required that credit is given to the original author. 2. Non Commercial (NC): The work cannot be used for commercial purposes. 3. No Derivative Works (ND): The work cannot be altered, transformed; derivative works built upon this work are not permitted. 4. Share Alike (SA): It is allowed to alter and transform the work, and to prepare derivative works upon this work, as long as the resulting work is licensed to a license identical to this one.

Each represents a unique way of combining these four attributions. The author is free to choose from amongst the eleven licenses, and to decide which one suits the work's requirements.

In 2004, Creative Commons released its second version of CC licenses. Since the requirement of attribution has been greatly adopted by users of CC licenses, the attribution requirement has become the default, and there are only six CC licenses in the second version. The eleven licenses in the first version, however, have not been superseded, and are still available.

The CC licenses have been designed for all kinds of digital content except software projects, including art work, photographs, music and literary text. It does not deal with the open source issue, since all the sources in such work are transparent, and are not compiled into forms which cannot be perceived. Some of the CC licenses do not allow modification and might not be regarded as not "free". The CC licenses are successful, however, in spreading the idea of freedom and openness among the larger public, which might be unfamiliar with software development.

1.6 Open Source Business models

Free software is generally available at little to no cost and can result in permanently lower costs compared to proprietary software, as evidenced by free software becoming popular in third world countries. With free software, businesses have the freedom to fit the software to their specific needs by changing the software themselves or by hiring programmers to modify it for them. Free software often has no warranty, and more importantly, generally does not assign legal liability to anyone. However, warranties are permitted between any two parties upon the condition of the software and its usage. Such an agreement is made separately from the free software license.
Under the free software business model, free software vendors may charge a fee for distribution and offer pay support and software customization services. Proprietary software uses a different business model, where a customer of the proprietary software pays a fee for a license to use the software. This license may grant the customer the ability to configure some or no parts of the software themselves. Often some level of support is included in the purchase of proprietary software, but additional support services (especially for enterprise applications) are usually available for an additional fee. Some proprietary software vendors will also customize software for a fee.

Free software gives users the ability to cooperate with each other in enhancing and refining the programs they use. Free software is a pure public good rather than a private good. Companies that contribute to free software can increase commercial innovation amidst the void of patent cross licensing lawsuits.

However, with the increasing awareness and usage of Free/Open Source Software with emphasis on freedom rather than 'free of cost', the focus is turning on the business modalities. In general, the stress so far has been on charging the service while giving the product free. This, in fact, faces some difficulties as in countries like India the taxation on software service may be more than selling it as a product. However, the service-oriented model has been gaining ground as more users get into FOSS arena and need "FOSS-enabled" professionals to install, maintain and troubleshoot their application systems. Since an application often involves a varied set of FOSS packages, bundling them together to work in an integrated environment needs some expertise. While the Linux distributions, for instance, provide the basic OS, RDBMS, Web Server, etc., usable coherently, a generic user would like to seamlessly include the packages of interest. Companies like SpikeSource specialize in such bundling and servicing the user clients.

1.6.1 The Dual Licensing Model

As second-generation open source vendors, MySQL AB, Sleepycat Software and Trolltech AS make the majority of their revenue from selling software licenses. This license-based business model offers higher margins than services-based businesses. Historically, most open source companies have tried to make money by selling services and support.

The guiding principle behind dual licensing is “quid pro quo,” or a fair exchange. Under this model, vendors offer their products under both an open source license and a commercial license. This allows open source projects to use the software at no cost, which contributes to widespread use and testing of the software and the fast growth of a large installed user base. Companies redistributing the software as part of commercial products can also get the benefits of the open source software by purchasing a commercial license, which releases them from requirements to publish their source code. Commercially-licensed customers generate revenue for the open source vendors, which contributes to the rapid development of high-quality software. The Dual Licensing business model is essentially based on three crucial factors: knowledge, time, money. The ultimate user does not possess the knowledge, might not have the time, but can pay for the service.

1.6.2 Some real life examples of business in FOSS:

Mentioned below are a few of the better known examples of some successful F/OSS based companies with a global presence:

1. **Red Hat:** RHEL is 100% GPL software. Red Hat also offers the totally free Fedora Core Linux (a more cutting edge version for enthusiasts). There are also free "clones" of RHEL, like CentOS and White Box. Yet, Red Hat's subscriptions, and profits, for RHEL have done nothing but go up and up and up. That is because Red Hat offers value in the subscriptions, support, and services. Businesses are more than glad to pay Red Hat for the support, instead of getting a totally free clone, because they can rely on Red Hat for testing, compatibility, and support for when things go wrong. Red Hat has software developers on their payroll, and they are also able to leverage the efforts of open source developers, as well as other companies.

2. **MySQL:** MySQL is totally free for most users. There is also a "pay-for" version with extensions and support. MySQL has been very profitable. MySQL has paid developers.

3. **Trolltech**: Trolltech develops the QT GUI widget toolkit. QT is offered in a dual licensing scheme. The first license is GPL, and the resulting software built on the GPL version must also be GPL. The KDE desktop environment is a perfect example of this. But if the company or developer wants to use QT to build a proprietary product, they must pay Trolltech a license. JDEdwards ERP software and Adobe Acrobat/PhotShop are both good examples of proprietary software built on the proprietary QT license. Trolltech is very profitable and has experienced growth in every year of it's existence. Trolltech has paid developers, including the original creator of KDE, Matthias Ettrich.

4. **Jboss:** JBoss is the open source J2EE application server and related tools. JBoss, the product, is completely free and open source. JBoss the company makes it's money on support and services based on JBoss the product. JBoss the company pays it's developers, and leverages the efforts of open source developsers. JBoss is profitable. JBoss has Java developers on it's payroll.

5. **SugarCRM** : SugarCRM is an open source Customer Relationship Mangagement software package based on open source PHP and Apache. The company makes it's money on packaging, managed services, and support. They are profitable as well. Sugar has paid developers.

6. **Novell** : The formerly slowly sinking ship completely reinvigorated itself by purchasing SuSE Linux and Ximian (the creators of Gnome and Evolution). Novell has returned to profitablitity, and has turned it's business around. Their "partnership" agreement with Microsoft has brought in a new dimension to the Open Source paradigm with the FOSS community offering both positive and negative comments.They have also open sourced some of their former proprietary products. Novell has developers on their payroll, including Miguel De Icaza, the creator and project leader of Gnome, Gnumeric, Evolution, and Mono.

There are undoubtedly a much larger, but may be lesser known, number of businesses in this domain and their number is bound to surge with time as the visibility and acceptance of F/OSS solutions keep expanding.

In this chapter, the features of Free/Open Source Software and its potential in catering to enterprise applications have been discussed with applications to banking and e-governance sectors mentioned as illustrative examples. Open Standards, the various Licensing options in FOSS and Open Source Business Models were some of the other topics introduced and described. The remaining chapters of this book elaborate on many of the concepts and technologies introduced here.

Chapter 2

F/OSS Development Methodology and Tools

Free/Open Source Software (F/OSS) thrives on the collaborative software development approach wherein the members of the community – both developers and users – join hands to identify the requirements, develop the software, check for bugs and enable deployment. This chapter starts with a discussion on the structure and dynamics of the F/OSS community and how the interaction is sustained through mailing lists, chat, wiki and various messaging tools. The features related to integrated development environments are discussed with Eclipse as an example. The later sections deal with the version control mechanisms and issue tracking methodologies to monitor and integrate the inputs, and includes bug fixing and other development related issues. The final section deals with the aspect of documentation which is very important in the F/OSS environment that has globally scattered developers and users.

2.1 Collaborative Software Development

Mankind needs to socialize for survival. Things cannot be done alone and one often has to seek help from others. Collaboration also enhances the quality and enables faster completion of the tasks. Developing quality software is one such task that needs participation from more than one individual.

2.1.1 What is collaborative development?

In a software project, its not just the people who code (the developers) who contribute towards the project. There are lot of others like management personnel, stake holders, users, security advisors, marketing managers and system support executives. Thus, all the above said and many more have to be kept informed about the project, but in varying levels. Thus, we need to find a way to manage and share information available within a software project space, such that the information is available without any hinder to all those who are involved in the project, can be shared with one another, as well as can be updated or modified as and when required. Thus, we are in need of a complex collaboration technique to enable a sound and healthy software development project.

The development of world wide web or the Internet as a very successful means of interchanging and sharing information has enabled us to utilize it as a tool for collaboration and sharing. The nature of contents of the web to be accessible from any part of the world has lead to universal sharing of information possible. It also enables the collaboration of human resources dispersed across the globe. Hence, two people sitting on the opposite sides of the planet are able to share their ideas and work together. This results in a situation where the resources are not confined to a particular location or need to be brought to a particular location to get a job done. When the skill required to do a specific task is not found locally or is not economically feasible, then the task can be easily sourced out to an appropriate place where the job can be done and the end product be sent back. Thus, the sharing of work takes place not within a single workshop, but with various workshops placed across the globe.

2.1.2 Why Collaborative Software Development ?

Looking at the various aspects of a software development process, we can understand why we need to follow collaborative development methodology in order to improve the development process and achieve the target in a more economical and efficient manner.

Sharing information

A software project involves different kinds of information at various stages of development and need to be shared with different kinds of people. The pre-development informations such as targeted product, the type of users focused upon, implementation details of the project, scheduled time line of the project, stake holder information and information regarding the members involved in the project should be globally available. At the same time, code and documentation may not be of interest to the marketing people. Thus, these huge volumes of information need to be effectively organized and managed for smooth operation of the software development process.

Sharing ideas

When working as a team with a common objective, each member of the team has some idea to suggest. Thus, even in a small sized team the volume of ideas which gets collected is so large that it needs effective management. Therefore we need a method to inform our ideas to others, discuss about it and decide on implementing them or otherwise. This entire method needs to be dynamic as ideas change with time, especially when a dozen eyes are looking into it and commenting about it. This is possible in a collaborative development, where ideas are hosted on a common white board and facilities available for others to comment and discuss further about it. When things are decided, it is again posted in the white board and finalized.

Sharing skills and experiences

In any typical software development, there exists a team of developers with different skills and experiences. Each individual member of the team does a specific type of task based upon his/her skill and experience level. For example, one developer may be focusing on the data storage and management, another on the business logic development while the third takes care of the interface development. Thus, instead of the same developer taking care of all three tasks, three developers can collaborate and develop in lesser time duration with better quality.

Parallel development

The more important advantage is that these three people can work independently with respect to their tasks, still share their information and can be well informed of what the others are doing. Thus, collaborative development involves parallel development as well, thus fuelling the pace of the project many times. In a typical software project, it is not just few developers but a combination of team leads, managers, developers, testers, technical writers and a lot more. These people have to be kept informed of the latest development in the project for them to stay in track and go along with the project. Collaborative development helps in such a situation by keeping them informed of what is happening and enabling them to add their contribution dynamically without interfering in other's work.

Sharing work

But, the common problem when working as a team is that there is a conflict of ideas, interests and information. That is, two different people trying to do the same thing might do it in different ways. Though the target is common, the end contributions from two persons will be in conflict with each another. For instance, two developers coding the same feature working on a common scratch code might give different final products. Another scenario is when a developer Y starts working on a code developed by X. During the development time, if X modifies the original code, the final product from Y may be meaningless unless Y is kept informed about the changes and suitable corrections are made.

"A dozen eyes are better than one"

When we try to solve a problem as a single person, our ideas are restricted and we try to view the problem from a single angle. When more than one person is involved in solving the problem, more ideas are poured in, more solutions are offered and tried out, our solutions are refined and improved by others and finally we land up in a very good situation-- and that too in a very minimal time comparatively. Thus, collaborative development is based on the fact that a dozen eyes are better than one.

No person can develop a software without even a single bug and also every developer feels that his/her code is without bugs. But, when he/she puts this code before a few others and offers them a chance to test them, bugs are found by others which might have been missed otherwise. Also, some of them might also be able to find a solution to these bugs and offer them to the original developer. Thus, the task of checking for error and hitting a solution when found one is made easy and quicker in the collaborative development process.

2.1.3 F/OSS and Collaborative Development

The Free and Open Source way of developing software has the ideas of collaborative development built inherently into it. The entire open source movement's success can be attributed to free availability of information, open sharing of information and, in particular, collaboration of developers across the globe over the Internet to develop quality softwares. Thus, the free software model can be termed as the best example of the collaborative development model, its success and superiority over the other models of software development being primarily a consequence of this fact.

In the F/OSS world, the contributors to a software development process are wide spread, from different countries, cultures and communities. Their skills and experience vary widely. But still they use the Internet as their medium to communicate with one another using simple tools like email and messaging services, to share ideas and work together for a common aim. They try to solve the conflicts and contradictions using the same Internet medium and develop a quality software. The main quality of such a software is that it is not specific to a certain instance. As the contributors are from different parts of the world, their ideas also are of different flavors. So, rather than looking the problem through a key hole, they look the problem from different angles and try to find a solution which satisfies all their requirements. Thus a software developed using FOSS's collaborative process becomes globally applicable, sometimes with little modification to suit local requirements.

One surprising aspect of FOSS's collaborative development is that in most cases the contributors or the co-developers would not have seen each other, spoken with each other or interacted personally with each other. But still, they collaborate with one another, share their ideas and knowledge and develop a high quality solution.

Almost all F/OSS products are developed using collaborative development process and the tools used themselves support collaborative development. As a result of the entire community indulging in collaboration and sharing, the community has developed excellent tools to enable collaborative development. Hence FOSS offers a load of economical and efficient collaborative development tools which have been tried and tested by thousands of developers all over the world.

2.1.4 Features of Collaborative Development

A typical software collaboration model should have facilities to enable a healthy collaboration where the freedom and security of the individual participants are guaranteed as well as the process of collaboration is smooth and simple. The important features of software collaboration are:

- The information is available globally from a single point and can be accessed from any place, depending upon the scope of collaboration.
- Any member involved in the collaboration can add, update, modify and remove information, provided that they have required permission to do so.
- The non development members of the project can actively participate in discussions and making decisions.
- The implementation requirements are minimal and affordable, and do not serve as an additional burden to the development process.
- The collaborators can work independently, without much dependency on others, from any place and at any time.
- The members can keep track of the project and inform others about their work status in an easy way.
- All the information are kept dynamic as much as possible.
- Proper source control mechanism is implemented thus enabling hassle-free sharing of source code and all conflicts occurring due to concurrent access of the same code are resolved.
- Codes from two different developers, working independently from different places can be merged and integrated in a simple and error free manner.
- The members can interact with the other members either on personal basis using messengers or globally through a news-group like mechanism.
- The bugs in software code can be easily tracked and reported, and the debugging process can be effectively shared and managed.

2.1.5 Tools of Collaborative Software Development

Collaborative development to be implemented on a software development process needs the aid of many tools depending upon the scope and type of collaboration. There are certain basic tools which can be used by even a small team involved in software development, which can very well improve the development process. These tools help in providing the individuals with updated and dynamic information, help and assistance, as well as an efficient method of sharing work between one another. F/OSS softwares are available for almost all of these tools, thereby making an economical implementation of collaborative development possible. Some of them are described below.

1. Wiki

A Wiki is an on line tool to share information, in such a way that the information can be accessed, added, updated and deleted by any person. Wiki's main purpose is collaborative editing of textual information. Wiki's are very useful during the initial stages of a project, when there are lots of incoming ideas and suggestions from the members, when things need to be discussed in an open manner and thus decisions be made based upon them. Wiki is thus useful on developing content and policies in a collaborative manner. Wiki is basically an on line web page, with facilities to open up new pages and add contents to them. Members can then add more contents to existing pages or edit the contents which had been added my some other. Thus with time, contents get added, refined and evolved into a resource pool of information, and this process occurs with free and open contribution from all members involved.

2. Mailing list

Mailing lists are effective way to interact within a group when the volume of the group is between moderate to heavy. The basic idea of mailing lists originated from the erstwhile newsgroups, where there is a common mail address to which the members send the mail and the mails sent to this

address gets forwarded to all the members who have subscribed to the mailing list. Thus the mailing list serves as a single node which shares the mail address to it amongst all its members. Thereby instead of mailing each and every individual in a group or sending a copy to each one of them, we can just send a mail to the mailing list and it is ensured that all the members get a copy of it. In return, the members can comment or reply on our mail and these gets shared among all the members. It can be simply said as an "all-gets-all-mails" mailing system, with a single mail address as access method. This is very useful to seek help from the members and get solutions, as all the members can see the mail, reply if they know any solutions, while others are also keeping track of the replies. This process results in improvement of knowledge level of the entire community as things are shared with every subscribed member of a mailing list.

3. Messaging and Chat services

Wiki's and mailing list are useful for discussion and developing some content. For quick help and discussion we need chat services where a few members can assemble on line and discuss. This is very useful during production time, when we need immediate answers from a specific people or group. We may also need to be in constant touch with members working along with us and mails may seem to be a distraction. In such situations personal messaging services may be helpful. There are lot of Open source messaging and chat services. Internet Relay Chat or IRC is one kind where the FOSS community is available for help and guidance.

4. Integrated Development Environments

Each programmer uses their own tools and often this results in a problem when each one is using a different type of tool. Also, each programmer needs different types of tools to get his job done. In such cases, a standard set of tools put together may solve the problem as all the tools are available together in a single place and all the programmers can use a single type of tool. Integrated development environments also have features that help in maintaining the code using a source control mechanism, network sharing and a lot more. Also that most IDEs are cross-platform such that they are available for all Operating Systems and hence the code developed in one can be used with the other. Thus, the code developed is more portable and reusable. The fact that IDEs have most of the required tools been integrated to them, reduces the work of developers, making it easy to use these tools from a single place without moving much around.

5. Version Control system

When more then one person is working on a code, each person implements his own codes and modifies the existing codes. Thus when people are working parallel, it is important to maintain the integrity between the individual copies as well as maintain the main source stable and updated. Version control system takes its position for such a role, allowing more than one to access the code, implement changes in the files and make the changes available to others. This is done by creating a new version of the code, denoted by a revision number, whenever a change is being done in the source directory known as the repository. Thus, when some one adds something to a file in the repository a new revision of the code is produced. When someone else tries to modify the same file, first the incoming data is checked whether its source has been the latest available code or a previous one. If it is based on the recent updated code, then the changes are merged into the file. If it is not, then the person trying to implement the changes is informed that he was working with an older file and in the meantime somebody has changed the file contents. Thus, the second person can ensure that he is not breaking the integrity and make sure his changes are with respect to last updated revision. This allows more then one person to work on the same file, even work on the same piece of code without resulting in a conflict and corresponding breaking of the code.

6. Issue Tracking system

Every piece of code written can have bugs and in a software project, every piece of code is tested for bugs. When bugs are found they need to be reported with appropriate information about it, so that the developers can solve it. Also, the reporting mechanism has to be efficient that the bug is not missed from being solved. Thus, we require an issue tracking system or a bug tracking system to keep in track of the bugs being reported and proper remedial measures being taken to debug them. In a typical issue tracking system, anyone who has found a big in a software can report it after checking whether there is already a similar bug reported. If so, then he can add more weightage to the bug, bringing it to the immediate concern of developers. Else, report a new bug with adequate information to find how the bug occurred and what caused it. On the other end, whenever a bug is reported developers who are responsible for the software are informed of an arrival of a bug. They can view the bug report, use the information and verify the bug. They can also assign and reassign the bug to particular developer or another project. Thus, the bug is always kept in track. When the bug is solved, all the individuals who had reported the bug are informed that a solution has been reached so they can solve the problem at their ends as well. Therefore issue tracking system is an important component to any software development process, especially in a collaborative environment.

2.1.6 Challenges in implementing Collaborative Software Development

An implementation of a new system into an existing environment is a challenge and certain resistances has to be crossed and all the involved people's comfort been taken care of. Collaborative environment demands certain things from the people involved as well as the organization implementing it, for the collaboration to be really effective. There are certain known challenges or issues which are to be overcome for the success of collaborative development to be achieved and are mentioned as follows.

Technical Issues

- Highly stable and web centric system is a challenge for the engineers.
- Future scalability and flexibility of the system needs to be analyzed.
- As the information is to be kept on line, the security of the information is to be ensured. This is very important as the organizations intellectual property are going to be placed else where, globally accessible.
- A decent speed quality inter networking is required for the system to run smoothly, which is still a dream in many countries of the world.
- Use of proper tools which satisfy the developer requirements need to be ensured. This includes a proper and comfortable user interface for the people to interact as well as tools to obtain information easily.
- It should be ensured that the source of information stays on line round the clock and is accessible to all concerned parties. Thus, proper backup systems has to be developed to maintain the availability at all time.
- Proper support structures has to be implemented and relating privilege issues be sorted out.

Psychological Issues

- When a team who have been used to handle things within themselves is asked to drop down everything to a common server and hence share them, generally shows some initial resistance.
- When the source is going to be kept not at a local storage and people are allowed to view

and edit it, raises the issue of responsibility for the integrity.
- In most corporates, sharing of information between teams is not welcomed and hence proper measures have to be taken to ensure that there is no external interference or trespassing into a team's private space.

Business Issues

- Corporates have to alter their business model to suit the collaborative development process.
- The use of tools and softwares in the development environment also needs to be evaluated to suit collaborative development.
- It is also to be determined which information or source code is accessible to whom and who can modify them. In short, permissions and privileges has to be sorted out before implementing a collaborative development system.

2.1.7 Steps of Implementation

Introducing collaborative development into an existing software project or among people used to a different development methodology is not a single day process and has to slowly evolve, for the collaboration to be successful. The process should begin with introduction of certain ideas and practices associated with a typical collaborative environment, and systematically the targeted collaborative development process should be introduced.

1. Collaborative development is based upon a community and hence a community building measure within the development team and the implementing organization should be first carried out. This will ensure that people can work with one another as a community, sharing things with each other and mutually benefiting.

2. People should be introduced to concepts such as mailing list, wiki, version control systems and bug tracking system. They should be allowed to experiment with them for some time to get used to these news ways of interaction.

3. Point of frictions during the community building process needs to be analyzed and appropriate solutions found.

4. Similar collaborative development projects should be studies and a suitable model should be derived, which will suit the environment, the developers and the business principles of the organization.

5. All the parties involved in this process should be explained and made to understand the necessity and advantages of implementing a collaborative development. All assistance should be offered to them to overcome the known friction points.

6. The implementation can be done starting with one or few teams and slowly implementing them into others. This will ensure people see the benefits of collaboration and voluntarily embrace it.

7. Tools and softwares to be used under collaborative development should be decided, documented and announced to all involved in the process. They should be given enough time to experiment with the tools which they are not used to and hence improve their comfort level with them.

8. All collaborative practices should be properly defined, documented and made openly available for reference at any time.

9. The process should be continuously monitored, friction points found and the process be optimized for more efficient operation.

2.1.8 Examples of Collaborative Development

Source Forge

Sourceforge.net is the world's largest open source software development project which hosts around 100,000 open source projects. It serves as a centralized resource for managing projects, project related issues, communication between the developers and users, and finally as a repository for maintaining and hosting the code. The main idea is to display the open source projects in a web site open to all, thus encouraging people to browse through the project. This helps both the developers as well as the end users.

Developers are provided with an area to host their project globally and hence attract global user base. Also, other developers and people interested in contributing to open source projects can browse through the existing projects and upon finding a project of interest can join the development. When you do not find a project, you can start your own project and others join to develop the software in a collaborative manner. The end users, on the other hand, can browse through a large collection of projects to satisfy their requirements and can pickup the softwares of their interest and try it out. They can also report bugs and feature requests to the developers, thus contributing to the project as well. Thus, the project gets evolved and matures into a high quality software.

The facilities SourceForge.net offers to those hosting their projects are,

- Web space for hosting the project, including shell access, database services and VHost
- File release system
- Donation system
- Version control system
- Communication tools like mailing lists, forums, tracking system
- Publicity by offering screen shots, statistics, news and more.

Linux Kernel Development

The best example for a successful open source project under collaborative development is the Linux Kernel Project. Right from day1, the Linux kernel was put under collaborative development. The source code was made available to freely download, modify and distribute. Thus, more people started trying out the kernel in their own computers and started giving feed backs. A few of them also started reporting bugs and solution for reported bugs. As time proceeded, more people joined the development team. At the same time, the other contributor volume also grew along with the user base. The collaborative development made it possible for the Linux kernel to develop and reach fame as it is now, through contributions from millions of people around the globe. A lot of other open source projects started following the same collaborative principles that the Linux Kernel project followed, thereby reaching success through collaboration. Some examples are :

- ***Drupal,*** a content management platform
- ***Mailman***, GNU's "manage electronic mail discussion lists"
- ***MediaWiki***, a wiki package originally written for Wikipedia
- ***MoinMoin***, wiki system based on Python

- ***Mozilla*** internet browser
- **SourceForge**, the software behind Sourceforge.net

2.2 The F/OSS Community

F/OSS Community can be defined as a group of people having a common interest in FOSS. The FOSS community actively promotes the adoption and awareness of FOSS, targeting a wide spectrum of users. They attempt to establish a community in which information will be freely exchanged, so that we may further the understanding of open source and its implications outside the realm of software development.

At one point in time, software developers were just considered to be geeky computer scientists, hiding behind machines developing tonnes of codes to create robots and programmes that helped to revolutionise the business and industry. Today, their role has evolved into something far much bigger and stronger, their contribution have shifted from scientific research, benefiting business and industry only towards developing freely accessible, modifiable, researchable, distributable contribution for humanitarian benefit. This has led towards formulation of the FOSS movement, not governed or owned by a single body, but by various FOSS communities, worldwide residing in civil society organs, academia and research, public and private sectors of society.

What brings and binds the FOSS community together ?

The FOSS community formulates together around the concept of community contribution. Most FOSS developers get together because they believe in making the world a better place, treating each other with respect and that everyone in the community has something to contribute. These principles evolve into the basis of a common, a community of sharing, where everyone equally shares resources and contributes more resources back to help retain the common grounds continuing to make them sustainable and useful for future generations.

While contributing to a FOSS community, it is not necessary that all members should be writing pseudocode or programming software code, using the various available open source software development language and platforms. Instead they can also contribute in terms of using the software for their own and their community's benefit, inform problems that occur while its usage, carry out documentation, promotion of the software online as well as offline, conduct training including seminars and conferences for wide-spread adoption by end-users, distribute the software freely or even charge fees for services built around using the software. The opportunities available for contributing in one form or the other are immense and the scale at which these activities may rise to is sometimes unimaginable. An effective example is that of GNU/Linux that has evolved into a globally renowned Operating System in competition to other proprietary platforms.

The FOSS community works around the project that continuously provides new solutions, features, and ideas. The FOSS community or the development team benefits from the output highly skilled software developers from around the world, creating a knowledge product of unimaginable economic and social value without paying huge sums of salaries and production overheads while benefiting from supplementary services models, built around the project in terms of learning and monetary values.

FOSS model has the ability to multiply itself like a nuclear reaction. It spreads from one region to the other by word of mouth, project and community mailing lists and newsletters and other promotional activities until it hits mainstream media and is picked up by the commercial corporate media world and thus, becomes a hot usable and sellable product. The model has also proved promote and market itself wildly like the examples of the GNU/Linux, Apache, the FireFox? browser and many other successful FOSS projects.

2.3 F/OSS Community Dynamics

When a number of people collaborate to do a task, there are number of ways they can interact with one another. It depends upon the physical location of each of them and as a team, that the mode of interaction varies. Also, they can have more than one way of interaction depending upon the purpose. Small teams put up with in a building can use a local messenger or chat software to enable them interact with one another, in addition to physical interaction. When the interaction need to be more detailed, like making a discussion of a feature inclusion they can have discussion forums. As the volume of information interchanged increases, the need to move to other way of interaction like wikis and emails arises. Thus, the mode of interaction depends upon the location, purpose and volume of interaction.

The few common methods of interaction in a collaborative development environment are forums, web blogs, wikis, chat and messaging system and mailing lists. A few of them like chat and messenger system are highly interactive and live such that the members will get quick responses. Mailing lists are not that much quickly responsive, but the main advantage is that everybody on a mailing lists are aware of things being discussed. Thus it is a much open interaction. IRC are another type of chat system where people from all over the world participate. Wikis are useful collaborative content development where people add and edit contents posted by them and others. But wikis are not useful for lively discussions like that in chat or mailing lists.

2.3.1 Mailing Lists

Mailing lists is a collection of mail addresses to whom mails will be sent. All the mail ids are behind a common mailing list address and any mails sent to this address will be sent to all in the list. Just a single mail addressed to the mailing list is enough to be sent to multiple recipients. One can add his mail address to the mailing list by subscribing to the list, hence agreeing that he is ready to get all the mails sent to the list as well as that his mail addressed to the list will be sent to all others in the list.

Traditional mailing lists started for postal services but the development of electronic mail has made it more successful and brought mailing list to the lime light. Mailing lists can be of two types; first is a distribution list which is a one way mechanism of sending mails to a group of people. This is suitable for announcement lists where people need not send a reply to the list. This is also used for commercial and marketing purposes like advertising new products to existing customers. But, mailing lists for collaborative environments need to be the other type where subscribers can reply back to the mails sent to the list.Mailing list are a very good medium for medium to high volume interaction with a medium to high volume group.

In a collaborative environment different types of mailing lists exist. General mailing list are for common discussion where every one in an organization participate and the topic of discussion can be of wide scope. Considering software projects, mailing lists can exclusively for users, developers, testers, announcements, installation, support, documentation and commercial. User lists are for users to post their queries and get the solutions from the community. Developer mailing lists are exclusive for developer to discuss issues and topics related to the development process and are highly technical. Announcements lists are used to announce patches, releases and updates to the subscribers. Installation lists offers help in installing a software. Support and documentation lists are related to the additional support offered to the customers and documentation work for the project respectively.

How Mailing List works

The mailing lists works in a simple yet effective manner. The subscribers name and mail id are stored in a database, while the mailing list itself has a mail address to be used for mailing to the list. There is also another mail address from which the mails will be distributed to the lists. Sometimes both these mail addresses can be the same. Thus, when a mail is sent to the mailing list, it reads the address list from the database and forwards the mail to all the subscribers as though the original sender sent to all of them. Thus for a receiver, it is not the mailing list sending them the mail but the sender itself is mailing them directly, thus the receiver can mail the original sender if he want to interact with him. Optionally, the mails may be distributed with the 'reply-to' part of the mail pointing to the mailing list's mail address and hence it is ensured that all the replies will come to the mailing list.

Generally most mailing list offers two modes of mails. One is individual mails where each mail sent to the list is sent to the subscribers as individual mails. This will increase the number of mails a subscriber receives from the mailing list. if the subscriber do not want individual mails to fill his inbox, then he can subscribe for a 'digest' mail where the mails are collected in a specific time interval and mailed together as a bulk mail. The time interval may either vary between an day to a week, as well as it can be a volume or number limit like one digest mail for every 10 mails or 600KB.

Mailing list Etiquettes

Mailing lists are a way of interaction and when large number of people are interacting things may go mayhem. Thus, we need to have some kind of guidelines on what things can be talked about and how. This will be very useful for new users of the mailing lists not to ask questions in an annoying way to the existing subscribers. Also, following these etiquettes will improve the interaction amongst the mailing list members. some common etiquettes are as follows,

1. Do not ask personal questions to a mailing list, provided it is not a mailing list meant for such purpose. If a discussion need to be taken upon personal interest and with a few people in particular, please take the discussion off the list and mail then personally.

2. While quoting a previous mail(s), please remove the unwanted portion from the mail which is not related to your reply, as well as please write you reply in the bottom, next to the quoted portion you are referring to.

3. When replying to a previous mail, please do not top-post but only bottom-post. Top posting means adding your reply to the top of the mail while the addressed contents are in the bottom. Bottom-posting means the contents you are referring from the previous mails are on the top of your content (or your content is in the bottom of them).

4. When quoting multiple quotes, i.e. from already quoted text, please ensure that the quoting is meaningful. Generally a first level quoting is indicated with a ">" in the beginning of the line, while successive levels of quoting are indicated with additional ">" for each level. For example,

> Am the first mail
>> am the reply to it.
Now again I want to quote them both.

5. When posting to a mailing list, please use a understandable subject line such that it clearly describes what you are speaking about in as few words as possible.

6. Do not post messages in Capital letters, it is considered as shouting. 7. Always take care that the language is not abusive or doesn't hurt any specific individual or people.

8. Most mailing lists have a volume limit for each posts. So trim your posts that it fits within the allowed limit.

9. Never quite the entire mail you are replying to.

10. Most subscribers would have subscriber for messages in plain text, so think twice before posting some HTML content in your mails.

11. Never send attachments with your mail to the mailing list. Upload them somewhere and give the links.

12. When you are referring to an external source in your reply or post, please give the appropriate link of the referred source if possible.

13. Do not troll in the mailing list. Trolling is causing disruption in the normal activities of the mailing list by posting incorrect or absurd information.

14. Please use the prescribed language of the mailing list.

15. The important point among the etiquettes is that before posting a question please do some home work in searching for relevant solution in the Internet. In your mail, please clearly specify what you intend to do, what you tried and what were the issues encountered.

16. Do not post any commercial messages in a mailing list until they are explicitly permitted or the purpose of the mailing list is commercial.

How to join a Mailing list

Mailing list are run by some mailing list software which provides an web interface for subscribing to the list as well as managing existing subscriptions. To join a mailing list, all that is needed is a valid mail address. Optionally a name can be given for the mail address and in addition a password need to be specified in order to modify subscription settings later. Once the subscription in accepted, the subscriber can choose additional settings like mailing list mode and language.

Another way of subscribing is to send a mail to a specific 'request' address of the mailing list, if it maintains one. Mostly this mail need to be a plain mail with the subject being "subscribe" or similar statement. Sometimes, more secure mailing lists such as the Linux Kernel Mailing list, send a mail to the subscriber in the mailing address he specified and a key or id needs to be sent back as a reply to that mail, with the key being the only content of the mail body. This method varies between mailing lists and are properly described in the web site of the mailing list.

Some Popular Mailing list softwares

Mailman [Source: http://www.gnu.org/software/mailman/]

Mailman is a Free Software for managing electronic mailing lists and e-newsletters. Its a GNU

software and it is freely distributed under GNU GPL License. It is written in Python programming language with a little bit of C code for security. It is integrated with the web, making it easy for the users to manage their accounts and for the list administrators to manage the lists and subscriptions. Mailman supports built-in archiving, automatic bounce processing, content filtering, digest delivery, spam filters and many more useful features.

Majordomo [Source: http://en.wikipedia.org/wiki/Majordomo_%28software%29]

open source mailing list manager (MLM) developed by Brent Chapman of Great Circle Associates. It works in conjunction with sendmail on UNIX and related operating systems. It came into widespread use beginning in 1992, predating the popularity of the web browser, in an era when many people had email access but no WWW access. As a result, tasks such as subscribing and unsubscribing are handled by sending mail messages, a process that seems cumbersome today.

There are front-ends, such as MajorCool?, to provide a web interface. Many mailing lists are still run using majordomo. There is work being done to completely rewrite Majordomo with a focus on keeping the familiar email interface while greatly improving the web interface and other features. This rewrite is referred to as Majordomo 2 and is currently being used by the OpenBSD project among others.

The important features of majordomo are,

- Supports various types of lists, including moderated
- All list management activities are handled by email, so list owners don't need access to Majordomo server machine
- Supports archival and remote retrieval of messages
- Supports digests
- Modular design - use only the features you need
- Written in Perl - easily customizable and expandable
- Includes support for FTPMAIL
- Supports confirmation of subscriptions, to protect against forged subscription requests
- List filters, based on header or body regular expressions

2.3.2 Wiki

Wiki is a collaborative content development system where users are free to add content, edit content which is already been added by some other, thereby evolving the content into a high quality information. Thus, wiki becomes a collection or resource center of open content.

Wiki is defined by its founder as follows, """ Wiki is a piece of server software that allows users to freely create and edit Web page content using any Web browser. Wiki supports hyperlinks and has a simple text syntax for creating new pages and cross-links between internal pages on the fly.

Wiki is unusual among group communication mechanisms in that it allows the organization of contributions to be edited in addition to the content itself.

Like many simple concepts, "open editing" has some profound and subtle effects on Wiki usage. Allowing everyday users to create and edit any page in a Web site is exciting in that it encourages democratic use of the Web and promotes content composition by nontechnical users. """

The first wiki to be launched was wikiwikiweb by Ward Cunningham during the period 1994-1995. The name comes from the Hawaiian word wiki-wiki for quick-quick. The names is also said to mean "what i know is.." symbolizing open contribution and sharing of knowledge.

How is a Wiki page constructed

Wiki's main idea is to allow open management of contents such that it receives contribution from various sources. This makes the content to be diverse and rich. Wiki uses a simple markup language to format the contents. All we need to use wiki is a simple web browser. Each page in wiki is known as "wiki page". Each page is interconnected via hyperlinks and this complex interconnection of individual pages actually forms the wiki. Thus it is nothing but a collection of vast number of pages created independently and linked with one another.

Generally the pages which are created or modified are not subjected to moderation and even there are open wikis which do not need a prior registration. Edits made to wiki pages mostly take immediate effect on line. But due to increased abuse of wiki pages, user authentication system are prescribed to keep off spam and abusive content from entering into the wiki pages.

Considering the format of a wiki page text, it makes use of a simple markup language to represent structural and visual specifications. This is like using simple symbols like '=' and '*' to implement a visual effect over a piece of text other than the markups, wiki page contains plain text.

In editing mode, the text appears as pure text content decorated with the markup symbols. When the editing is done and the page is published, the markups spring into action converting the symbols into visual effects like bold or underlined text, bulleted list, linked url and many more.

Upon each edit, an edit summary can be added. This is similar to a log string used in version management. This string is displayed when edit history is browsed. Most wikis support revision control and hence the use of edit string is quite obvious.

The one difference that wiki page has from a HTML page is that wiki pages do not use HTML tags, or even JavaScript? and CSSS. But, it uses its own markup language to bring a similar look and feel of a web page.

There are front end wiki tools available which offers a WYSIWYG (What You See Is What You Get) editors, which make use of JavaScript? and Ajax to convert the visual effects from the editor into a proper wiki markup. This is useful when the user is not comfortable with the wiki syntaxes and prefers an easy tool to do the editing.

How to use a Wiki

Wiki pages can be used to put up content, which can be edited by any one. To create a wiki page, we need a basic knowledge of some wiki markup syntax. These syntax are used to add some effect to the otherwise plain text, thus giving in a nice look and feel similar to a web page.

Wiki either have a create page button to create new pages or we can use the existing pages to create links to new pages and thereby create them dynamically. Any link which doesn't point to any existing page can be used to create a new page. Most wikis implement the CamelCase naming for page identifiers. That is, any CamelCased? string will be considered as a page link. If there is no existing page with the name same as the string, a new page can be created by following the link.

Similarly, we can just link a text to another page by making it a camel case. Hence, be careful with the text entered, CamelCase text can confuse people as they are displayed like a page link.

A few common syntax of wiki markup are given below.

```
''' bold '''
'' italic ''
__underline__
^superscript^
,,subscript,,
```

bold *italic* underline superscript subscript

```
= Heading =
== Subheading ==
=== About ===
==== Paragraphs ====
```

Heading ¶

Subheading ¶

About ¶

Paragraphs ¶

```
 * item 1
   * item 1.1
 * item 2
```

- item 1
  - item 1.1
- item 2

```
|| cell 1,1 || cell 1,2 ||
|| cell 2,1 || cell 2,2 ||
```

cell 1,1	cell 1,2
cell 2,1	cell 2,2

```
Title Link, http://www.myindex.com
```

Title Link, http://www.myindex.com

```
[http://www.google.com Google.com]
```

Google.com

In addition to these syntax, the various wiki softwares offers their own syntax for links and macros.

Such syntax are very useful in creating links from parent to child pages and from child pages back to the parent pages.

Misuse of Wikis

Anything which is available free and open to all are subjective to vandalism and misuse. Wiki too isn't an exception for this, as the rule is that any one can edit or change content. But, off late this has been taken up seriously and counter measures have been included in the wiki system itself. Measures like user authentication, edit or delete only by privileged users, moderation of contents, constant monitoring of content change in a wiki system and many more have really tried to drive off the offenders of this wonderful open tool.

[Source: http://en.wikipedia.org/wiki/Wiki]

Some Popular Wiki softwares

Java Based

- JSPWiki is a J2EE based wiki application under the GPL

JavaScript? Based

- Tiddlywiki is a HTML/JS based server-less wiki where the entire site is contained in a single file.

Perl Based

- Twiki is a structured wiki used for project development applications and has a document management system and a groupware tool.
- wikiwikiweb by Ward Cunningham, the predecessor to the wikis.

PHP Based

- MediaWiki? is a high-volume capable wiki, which was used for Wikipedia. It uses MySQL or PostgreSQL as database back-end. It is publicly available and can be customized to suit our application.
- TikiWiki? is a large wiki development project with a variety of groupware tools.
- PhpWiki? is a clone of wikiwikiweb in PHP

Python Based

- Trac is a wiki clone integrated with a issue tracking system and interface to Subversion for version control.
- MainMain? is a wiki clone in Python offering good access control using user groups.

[Source : http://en.wikipedia.org/wiki/List_of_wiki_software]

Media Wiki

MediaWiki? is a web-based wiki software application used by all projects of the Wikimedia Foundation, all wikis hosted by Wikia, and many other wikis, including some of the largest and most popular ones.[1] Originally developed to serve the needs of the free content Wikipedia encyclopedia, today it has also been deployed by companies as an internal knowledge management solution, and as a content management system. Notably, Novell uses it to operate several of its high traffic websites, which are not editable by the general public.[2]

MediaWiki? is written in the PHP programming language, and can use either the MySQL or PostgreSQL relational database management system. MediaWiki? is distributed under the terms of the GNU General Public License, making it free software

Key Features of MediaWiki?

- Links - MediaWiki? doesn't follow the CamelCase syntax to create links, instead uses a markup of [[ ]] to create links. Anything surrounded by this markup becomes a link to a page. Thus, awkward use of camel case within content can be eliminated.

- Namespaces - These are prefixes which can be used to create page with the same name but with different purposes. The name spaces are prefixed to the page titles. A few common name spaces are

- Media: - Upload media files directly
- User: - User profiles
- Project: - Project Information
- Category: - Category of articles
- Template: - Reusable blocks of information

- Subpages - This feature provides a back link to the main page.

- Categories - These are user created and similar to tags, which help in organizing information based upon categorization.

- Templates - These are dynamically loadable text blocks which gets added into the page whenever it is loaded.

- Graphical tools bar to ease the learning of wiki syntaxes.

- Editing subsections instead of opening the entire page for editing.

- Rich content support and support for LaTeX and OCaml.

- Customizable interface.

- Extensions to hook in additional code offering extra features.

[Source : http://en.wikipedia.org/wiki/MediaWiki]

Installing Media Wiki

MediaWiki? software can be downloaded from MediaWiki? home page

(http://www.mediawiki.org). In addition to the software, the requirements include a Web Server (Apache), a Database server such as MySQL or PostgreSQL, PHP version 5.0 or later. Additional softwares such as PhpMyAdmin? to ease the installation softwares are also supported.

MediaWiki?, by itself, comes along with an installation script which can be executed in a web browse. This script will take us across the installation process, asking us a few questions and thereby installing the software at proper location. The database can be shared with other PHP based application such as phpBB as well.

In order to customize MediaWiki? for personal preferences and requirements, knowledge of PHP is required. The installation requires a existing database with a username and password for it. The installation is a simple step by step process involving setting up of the database and installing the software at appropriate location, both of which is taken care by the installer. Any errors that occur during the installation can be solved by referring to the installation guide present in the MediaWiki? home web site.

[Source : http://meta.wikimedia.org/wiki/Help:Installation]

2.3.3 Chat and Messaging

Although Wiki and Mailing lists may solve the problem of interaction and sharing of information, they are not instant way to interact. For example, posting a mail to a mailing list may take an hour to a day to a week to get any reply. For needs of instant interaction an instant messenger is needed.

Instant messaging is a form of quick interaction and exchange of information between two or a few people. Internet is used as a medium of interconnection, though it can also be possible over an intra net or a local LAN.

Instant messaging requires the use of a client program that hooks up an instant messaging service and differs from e-mail in that conversations are then able to happen in real-time. Most services offer a presence information feature, indicating whether people on one's list of contacts are currently on line and available to chat. This may be called a contact list. In early instant messaging programs, each letter appeared as it was typed, and when letters were deleted to correct typos this was also seen in real time. This made it more like a telephone conversation than exchanging letters. In modern instant messaging programs, the other party in the conversation generally only sees each line of text right after a new line is started. Most instant messaging applications also include the ability to set a status message, roughly analogous to the message on a telephone answering machine.

Though there are tons of commercial instant messaging services available, we will rather be looking for a non-commercial instant messaging system for using within a collaborative environment. In FOSS, there are a quite famous instant messaging systems. Two among them are Jabber and Internet Relay Chat.

Instant Messaging with Jabber

Jabber is an open system primarily built to provide instant messaging service and presence information (aka buddy lists). The protocol is built to be extensible and other features such as Voice over IP and file transfers have been added.

Unlike most instant messaging protocols, Jabber is based on open standards. Like e-mail, it is an open system where anyone who has a domain name and a suitable Internet connection can run their own Jabber server and talk to users on other servers. The standard server implementations and many clients are also free/open source software.

Jabber-based software is deployed on thousands of servers across the Internet and even Google uses Jabber as a backend for it GTalk Instant Messenger service.

Features of Jabber

The main features of Jabber serves as an advantage of it over other commercial and non-free instant messaging systems.

- Open -- the Jabber protocols are free, open, public, and easily understandable; in addition, multiple implementations exist for clients, servers, components, and code libraries.
- Standard -- the Internet Engineering Task Force (IETF) has formalized the core XML streaming protocols as an approved instant messaging and presence technology under the name of XMPP, and the XMPP specifications have been published as RFC 3920 and RFC 3921.
- Proven -- the first Jabber technologies were developed by Jeremie Miller in 1998 and are now quite stable; hundreds of developers are working on Jabber technologies, there are tens of thousands of Jabber servers running on the Internet today, and millions of people use Jabber for IM.
- Decentralized -- the architecture of the Jabber network is similar to email; as a result, anyone can run their own Jabber server, enabling individuals and organizations to take control of their IM experience.
- Secure -- any Jabber server may be isolated from the public Jabber network (e.g., on a company intra net), and robust security using SASL and TLS has been built into the core XMPP specifications.
- Extensible -- using the power of XML namespaces, anyone can build custom functionality on top of the core protocols; to maintain interoperability, common extensions are managed by the Jabber Software Foundation.
- Flexible -- Jabber applications beyond IM include network management, content syndication, collaboration tools, file sharing, gaming, and remote systems monitoring.
- Diverse -- a wide range of companies and open-source projects use the Jabber protocols to build and deploy real-time applications and services; you will never get "locked in" when you use Jabber technologies.

2.3.4 Internet Relay Chat

Internet Relay Chat (IRC) is a form of real-time Internet chat or synchronous conferencing. It is mainly designed for group (many-to-many) communication in discussion forums called channels, but also allows one-to-one communication and data transfers via private message

Protocol

The client-server protocol being used now is a descendant of irc2.8 version protocol which is documented in RFC 1459. Though there exists many specifications on IRC protocols, there is no official specification and hence all specifications are dynamic that they change as technology improves and implementation differs. The important point to note is the client-to-server protocols are almost identical with one another. But this is not true on the server side, thus making the linking

of two different implementations of IRC server very cumbersome. This problem is tackled by implementing bridge servers but this is not widely deployed due to implementation issues.

Using IRC

To use IRC and join the network, an IRC client is required. This is similar to a instant messenger client, but has additional and different features to suit the IRC. Many instant messenger clients like Gaim have support for IRC as well. In addition, one need to select a nick which uniquely identifies one on the network. This nick can be registered with the IRC server and this becomes unique to that person. Nicks are protected by passwords and hence when a person's registered nick used by some other, he can retrieve his nick by identifying himself with the password. By default IRC clients do not get into any specific channel and hence the user needs to find some channels of his interest. But, IRC clients have facilities to list all channels related to a term or topic. This can also be done using the /list command. When joined into a channel, messages can be directly types into the typing area and the message appears is visible to all the users in that channel.

Common IRC Terminologies

Channel - Channels are more like chat rooms which can host a number of users and each channel is targeted at some purpose. Users can login to multiple channels and post their messages which become visible on the IRC client of the users in the channel. Channels on a IRC network are denoted with a "#" before their name while channels local to server are denoted with a "@". Channels have various permissions attached with them such as moderation, topic, secrecy, private channels, etc. A user can join a channel issuing a */join channel_name* command in their IRC client.

Mode - Modes are like parameters of a user or a channel. Thus, we have user modes and channel modes. Modes are represented by single letter symbols and are set using the */mode* command. Modes are basically permissions associated with an user or a channel. Typical channel modes are private(P), secret(S), invite only(I), no outside messages(N), topic locked(T) and moderated(M).

Ops or Operators - Operators are like moderators who have administrative privileges over channel. The Ops have addition privileges of allowing people into a private channel, banning users, removing abusive users temporarily from the channel, maintaining topic of a channel and more. Operators have some guidelines to follow in maintaining the channel and are described by the corresponding IRC networks.

Nick - Nicks are unique names owned by users which are used to identify them on the IRC network. Nicks are valid across channels and hence are not specific to a channel. Nicks help in identifying a specific users from one another and especially from anonymous users. If one wants to own a nick, he has to register the nick with a password. This can be done with the */register* command.

ChanServ? - This stands for Channel Services, which are the service servers of the IRC network for channel related operations. Channel maintenance are responsibility of the channel server and any command related to channel are to be sent to the ChanServ?.

NickServ?' *- This stands for Nick Services which are service servers associated with nick or users. Any user related command has to be sent to this server. One good example is identification of a nick by issuing its password to the NickServ?.*

[Source : http://en.wikipedia.org/wiki/Internet_Relay_Chat]

IRC Etiquettes

- Do not type in Capital case.
- Do not repeat yourself when asking something, this will annoy people resulting in getting ignored.
- Do not flood the channel with long text or copying something huge from the Internet. Rather paste it to a service like paste bin or give the corresponding link.
- Use correct terminologies of IRC networks.
- Do not abuse people or speak personal matters.
- Please stick to the intended topic of the channel, do not digress too much.
- When you are having problem with people in the channel, seek the help of the Ops.
- When a single person in the channel or network is troubling you, put him on ignore using the /ignore command.
- Do not use SMS speak in a channel.
- Do not send a personal message to any user without obtaining prior permission from them.

2.4 Integrated Development Environments (IDEs)

IDE or Integrated Development Environments are a certain type of computer software aimed at easing the software development process encountered by developers. Generally, any software development consists of multiple process which includes writing the code, editing, compilation, debugging and testing. For each process a separate tool might be used and thus the work needs to be moved between one tool to another.

Considering the time limit a software developer has to complete his software, a lot of time gets wasted from moving the code between one tool to another. Also, this tool chain may not be having tools which are perfectly compatible with one another. So, there arises an additional task of converting the output of one tool into a format which the other tools can understand. Also, all these tools need to be configured as per the project requirements. Thus, the project development process become cumbersome while using bits and pieces of tools.

A solution to this problem is integration of tools which are commonly used for a specific kind of software development. This tools are both physically and programmatically integrated such that the entire process can be finished with a few mouse clicks. Also, the presence of all the required tools within a single location makes things easier as the developer need to take the trouble to find where the tool is or where the output of the last tool was kept.

In addition to being together, it also ensured that the tools can perfectly inter-operate with one another such the code can be easily moved from an editor to a compiler and then to a debugger if needed. In most IDEs we can just enter the code in the editor, save it and use the inbuilt compiler or interpreter to work on this code. And when the code is being built, the errors and status are shown in a separate status box. Thus, the debugger can operate on these errors and it marks the corresponding lines of code in the editor itself. Thus, the job of the developer in finding where the error is present in the code it removed and the productivity of the developer improves.

It is also have become possible that the same IDE can build the code for different configurations with almost the same code. Thus the work load on the developer is still further reduced and hence softwares can be developer faster and in a much more efficient way.

Modern IDEs also include tools to visually represent the code in the form of flow or function diagrams and hence the relation between one part of the code and another part can be easily mapped and understood. This is also helpful in finding errors in the logic which makes the code to behave in an entirely different way than we want it to.

2.4.1 Common features offered by IDEs

- Code completion - Most IDE can help with code completion, where entering a part of the standard syntax and pressing a tab will complete the rest of the code. Alternatively, a drop down list of possible code completions might be shown from which we can pick the appropriate code.

- Code insight - Modern IDEs, when specified with the programming language being currently handled, is aware of the knowledge constraints, idioms and syntaxes that it can highlight certain parts of the code with unique colors. This will help us in finding errors in declarations and syntaxes that is made during typing. In addition, it can also mark open and closed braces and other grouping syntaxes.

- Resource Management - A program requires additional resources like libraries in order to execute the code. An IDE can help in ensuring all such resources are available and if not give a warning or error during compilation so the issue can be solved at the stage of development itself.

- Debugging - IDEs include an internal debugger to debug our code. Debugger can test the code and help in debugging the bugs in them. Most IDEs offer integration with commonly used debuggers like gdb.

- Compile and Build - The compilation and build process varies between programming languages and hence an IDE include appropriate compiler and build tools for each and every programming language supported by the IDE. It might also offer facility to integrate external tools along with the IDE through plugins.

2.4.2 Advantages of using an IDE

- The prime purpose of an IDE is to make the development process faster and this is what an IDE makes possible by integrating tools and features within them.

- When an entire team or project follows a particular IDE, then the implementation of standards and specifications within the project becomes easier and possible.

- Most IDEs come with additional tools to look after documentation, prepare reports based on tests run as well as the edit-compile-build time taken by the developer. Thus, the project management requirements are aided by the IDE.

2.4.3 Downsides of using an IDE

- For a developer who is used to making use of independent tools, there lies a steep learning curve to learn the new tools that comes along with an IDE, find how to make use of the features of IDE and important how to access the features and tools provided by the IDE.

- When programmers start to learn programming with an IDE which provides every kind of tool and they get used to it, they become addicted to that particular IDE and hence may not be efficient in using any other tool or IDE in the future. Also, use of IDE may not make the programmer realize how the tool actually does the process.

- IDEs might improve the productivity of the user but it may not improve the proficiency. IDEs are tools which can be handled by a proficient programmer, like how a paint programmer can be handled by a creative designer. IDE should be merely considered as a tool and not as a solution for programming problems.

2.4.4 Some Popular Open Source IDEs

Eclipse is an open-source platform-independent integrated development environment. Although it was originally written for Java programming development, it has supports added to other programming languages such as C++ and Python. It is one of the best and widely used IDE for Java programming and development, complete with the Java Development Toolkit and the Compiler. Eclipse will be dealt with in more detail in the next sub-section.

KDevelop is a free IDE for GNU/Linux and other Unix-like operating systems. KDevelop 3.0 was a complete rewrite of KDevelop 2. It was released together with KDE 3.2 in February 2004. KDevelop is licensed under the GPL license. KDevelop does not include a compiler; instead, it uses the GNU Compiler Collection (or, optionally, another compiler) to produce executable code. The current version, 3.4, supports many programming languages such as Ada, Bash, C, C++, Fortran, Java, Pascal, Perl, PHP, Python, Ruby, and SQL

Anjuta is an integrated development environment for C and C++ on GNU/Linux. It has been written for GTK+/GNOME and features a number of advanced programming facilities. These include project management, application wizards, an interactive debugger, and a powerful source code editor with source browsing and syntax highlighting

GNU Emacs is described in the GNU Emacs manual as "the extensible, customizable, self-documenting, real-time display editor." It is also the most portable and ported of the implementations of Emacs. GNU Emacs is a Free Software text editor with a extensive collection of features which can help in development process of most programming languages.

Vim stands for Vi IMproved, and is an open source, multi-platform text editor extended from vi. It was first released by Bram Moolenaar in 1991. Since then, numerous features have been added to Vim, many of which are helpful in editing program source code. Vim and vi are the most popular editors for programmers and users of Unix-like operating systems.

Quanta Plus is a free software web development environment for KDE. Quanta is capable of both WYSIWYG design and handcoding.

Eric3 is a free Integrated Development Environment for the Python programming language. It is written using the PyQt? Python bindings for the Qt GUI toolkit and the QScintilla editor widget.

Gambas, under the GPL licence, is based on a BASIC interpreter with object extensions. It is designed to be a programming language like Visual Basic that can run under Linux.

2.4.5 Eclipse IDE

Eclipse in an open-source platform-independent IDE which has complete support for Java and C++ development environments as well as many other programming languages like Python. Eclipse, along with its tools, has proved itself to be a cost effective alternative for making the development process easy and faster.

Plugins like JBOSS-IDE, Lomboz based J2EE development environment and MyEclipse? plugin for J2EE have improved the use of Eclipse as a standard IDE for many development environments.

Downloading and Installing

Eclipse is free to download and distribute, being under an Open Source License. It can be downloaded from the official site at http://www.eclipse.com.

Eclipse is characterized by a very simple and hassle-less installation procedure. Eclipse is available in both .zip and .tar.gz formats. What is to be done is to extract this compressed file into a chosen location. We can just move into the location and clik on the Eclipse icon. Alternatively when using a terminal, make the folder as the current directory and execute Eclipse with the command "./eclipse" in the terminal.

Workspace

Workspace is the primary working area in the development environment provided by Eclipse. This is where the IDE physically builds it projects. The workspace folder contains all the source folders and files, complied files and many other project related information.

When the Eclipse IDE is started, a dialog box opens up asking for the workspace for the current project to be specified.

Perspective

Many types of GUI styles are available in Eclipse which is tailored to satisfy certain requirements and developer models. Each GUI style is called a Perspective and it facilitates developers in making their projects. A perspective includes a development window placed in the center, package and project explorers to the sides and windows for tasks and problems below.

The default perspective in Eclipse is Java perspective. A different perspective can be selected using the menu, Window -> Open -> Perspective -> Other and selecting one from the available perspectives.

Managing Projects in Eclipse

Environments

The environment settings for each perspective can be managed by using the property window, accessed by the menu Window --> Preferences and selecting the appropriate environment. For example, Java environment can be selected from preferences followed by Installed JREs to modify environment settings for any of the installed JREs. Other important setting is the CVS settings in the preferences window.

Creating Projects

New project can be started by selecting File -> New -> Project and selecting the corresponding

project such as Java project or C++ project. Then the project orientation wizard will guide through the step-by-step process of creating a new project. During this process, we need to give a project name and specify project workspace (or create new). We can also specify the libraries to be used for the current project and can also add new libraries. When we finish and click the Finish button, a new project comes into life.

Editing code

When working with Java, we can create new class by selecting New -> Java -> Class. This opens up a new Java class wizard. Upon giving the required information, a new class is created and available in the editor. Instead we can directly type our code into the editor.

Compile and Build

By default, when we save our code in the editor Eclipse tries to compile and build the code. This can be turned on and off using the Project -> Build Automatically option. To manually build a project, select Project -> Build All.

Running an application

To run an application, select Run -> Run As -> type of application.

This runs the application and prints the output into the console window placed below.

Getting Help

Eclipse comes with an in-built Help feature available under 'Help' Menu.

2.5 Version Control Systems

Version control, also known as Revision control or Source code management (SCM), is the management of multiple revisions of the same unit of information. It is most commonly used in engineering and software development to manage ongoing development of digital documents like application source code, art resources such as blueprints or electronic models and other critical information that may be worked on by a team of people. Changes to these documents are identified by incrementing an associated number or letter code, termed the revision number, revision level, or simply revision and associated historically with the person making the change. A simple form of revision control, for example, has the initial issue of a drawing assigned the revision number 1. When the first change is made, the revision number is incremented to 2 and so on.

Revision control is any practice that tracks and provides control over changes to source code. Software developers sometimes use revision control software to maintain documentation and configuration files as well as source code. In theory, revision control can be applied to any type of information record. However, in practice, the more sophisticated techniques and tools for revision control have rarely been used outside of software development circles.

2.5.1 Why Version Control ?

As software is developed and deployed, it is extremely common for multiple versions of the same software to be deployed in different sites, and for the software's developers to be working simultaneously on updates. Bugs and other issues with software are often only present in certain versions (because of the fixing of some problems and the introduction of others as the program develops). Therefore, for the purposes of locating and fixing bugs, it is vitally important to be able to retrieve and run different versions of the software to determine in which version(s) the problem occurs. It may also be necessary to develop two versions of the software concurrently (for instance, where one version has bugs fixed, but no new features, while the other version is where new features are worked on).

At the simplest level, developers could simply retain multiple copies of the different versions of the program, and number them appropriately. This simple approach has been used on many large software projects. While this method can work, it is inefficient as many near-identical copies of the program have to be maintained. This requires a lot of self-discipline on the part of developers, and often leads to mistakes. Consequently, systems to automate some or all of the revision control process have been developed.

2.5.2 Storage Models

File locking

The simplest method of preventing concurrent access problems is to lock files so that only one developer at a time has write access to the central repository copies of those files. Once one developer checks out a file, others can read that file, but no one else is allowed to change that file until that developer checks in the updated version (or cancels the checkout).

Version merging

Most version control systems, such as CVS and Subversion, allow multiple developers to be editing the same file at the same time. The first developer to check in changes to the central repository always succeeds. The system provides facilities to merge changes into the central repository, so the improvements from the first developer are preserved when the other programmers check in.

The concept of a reserved edit can provide an optional means to explicitly lock a file for exclusive write access, even though a merging capability exists.

Distributed version control

Distributed systems inherently allow multiple simultaneous editing. In a distributed revision control model, there is no such thing as checking in or out. Instead, every programmer has a working copy that includes the complete repository. All changes are distributed by merging (pushing/pulling) between repositories. This mode of operation allows developers to work without a network connection, and it also allows developers full revision control capabilities without requiring permissions to be granted by a central authority. One of the leading proponents of distributed revision control is Linus Torvalds, the main developer of the Linux kernel. He made the GIT distributed version control now being used by the Linux kernel developers.

2.5.3 Some Popular Version Control Systems

CVS

CVS (Concurrent Version System) is extremely popular, and when it was released, CVS was a major new innovation in software configuration management. However, CVS is now showing its age through a number of awkward limitations: changes are tracked per-file instead of per-change, commits aren't atomic, renaming files and directories is awkward, and its branching limitations mean that you'd better faithfully tag things or there'll be trouble later. Some of the maintainers of the original CVS have declared that the CVS code has become too crusty to effectively maintain. These problems led the main CVS developers to start afresh and create Subversion.

SVN

Subversion (SVN) is a new system, intending to be a simple replacement of CVS. Subversion is basically a re-implementation of CVS with its warts fixed, and it still works the same basic way (supporting a centralized repository). Like CVS, subversion by itself is intended to support a centralized repository for developers and doesn't handle decentralized development well; the svn project extends subversion to support decentralized development.

GNU Arch

GNU arch is a very interesting competitor, and works in a completely different way from CVS and Subversion. GNU Arch is released under the GNU GPL. GNU arch is fully decentralized, which makes it very work well for decentralized development (like the Linux kernel's development process). It has a very clever and remarkably simple approach to handling data, so it works very easily with many other tools. The smarts are in the client tools, not the server, so a simple secure ftp site or shared directory can serve as the repository, an intriguing capability for such a powerful SCM system. It has simple dependencies, so it's easy to set up too.

Bazaar

Bazaar is a decentralized revision control system. Revision control involves keeping track of changes in software source code or similar information, and helping people work on it in teams. Bazaar is a free software project with a large community of contributors, sponsored by Canonical Limited, the founders of Ubuntu and Launchpad. Bazaar is genuinely Free Software, released under the GNU GPL. It is written in Python and designed for correctness, performance, simplicity, and familiarity for developers migrating from CVS or Subversion.

Bazaar branches can be hosted on any web server, and uploaded over sftp, ftp, or rsync. For the fastest possible network performance, there is a smart server. Bazaar supports flexible work models: centralized like cvs or svn, commit offline, enforced code review when desired, and automatic regression testing. Decentralized revision control systems give people the ability to collaborate more efficiently over the internet using the bazaar development model. Using Bazaar, commit can be done to our local branches of our favorite free software projects without needing special permission.

2.5.4 Common Terminologies in Version Control

Repository

The repository is where the current and historical file data is stored, often on a server. Sometimes also called a depot.

Working copy

The working copy is the local copy of files from a repository, at a specific time or revision. All work done to the files in a repository is initially done on a working copy, hence the name. Conceptually, it is a sandbox.

Check-out

A check-out (or checkout or co) creates a local working copy from the repository. Either a specific revision is specified, or the latest is obtained.

Commit

A commit (check-in, ci or, more rarely, install or submit) occurs when a copy of the changes made to the working copy is written or merged into the repository.

Change

A change (or diff, or delta) represents a specific modification to a document under version control. The granularity of the modification considered a change varies between version control systems.

Change list

On many version control systems with atomic multi-change commits, a changelist, change set, or patch identifies the set of changes made in a single commit. This can also represent a sequential view of the source code, allowing source to be examined as of any particular changelist ID.

Update

An update (or sync) merges changes that have been made in the repository (e.g. by other people) into the local working copy.

Branch

A set of files under version control may be branched or forked at a point in time so that, from that time forward, two copies of those files may be developed at different speeds or in different ways independently of the other.

Merge

A merge or integration brings together two sets of changes to a file or set of files into a unified revision of that file or files.

- This may happen when one user, working on those files, updates their working copy with changes made, and checked into the repository, by other users. Conversely, this same process may happen in the repository when a user tries to check-in their changes.
- It may happen after a set of files has been branched, then a problem that existed before the branching is fixed in one branch and this fix needs merging into the other.
- It may happen after files have been branched, developed independently for a while and then are required to be merged back into a single unified trunk.

Dynamic stream

A stream (a data structure that implements a configuration of the elements in a particular repository) whose configuration changes over time, with new versions promoted from child workspaces and/or from other dynamic streams. It also inherits versions from its parent stream.

Revision

A revision or version is one version in a chain of changes.

Tag

A tag or release refers to an important snapshot in time, consistent across many files. These files at that point may all be tagged with a user-friendly, meaningful name or revision number.

Import

An import is the action of copying a local directory tree (that is not currently a working copy) into the repository for the first time.

Export

An export is similar to a check-out except that it creates a clean directory tree without the version control metadata used in a working copy. Often used prior to publishing the contents.

Conflict

A conflict occurs when two changes are made by different parties to the same document or place within a document. When the software is not intelligent enough to decide which change is 'correct', a user is required to resolve such a conflict.

Resolve

The act of user intervention to address a conflict between different changes to the same document.

Baseline

An approved revision of a document or source file from which subsequent changes can be made.

2.6 Subversion - An Introduction

Subversion is a free/open-source version control system. That is, Subversion manages files and directories over time. A tree of files is placed into a central repository. The repository is much like an ordinary file server, except that it remembers every change ever made to your files and directories. This allows you to recover older versions of your data, or examine the history of how your data changed. In this regard, many people think of a version control system as a sort of "time machine".

Subversion can access its repository across networks, which allows it to be used by people on different computers. At some level, the ability for various people to modify and manage the same set of data from their respective locations fosters collaboration. Progress can occur more quickly without a single conduit through which all modifications must occur. And because the work is versioned, you need not fear that quality is the trade-off for losing that conduit—if some incorrect change is made to the data, just undo that change.

Some version control systems are also software configuration management (SCM) systems. These systems are specifically tailored to manage trees of source code, and have many features that are specific to software development—such as natively understanding programming languages, or supplying tools for building software. Subversion, however, is not one of these systems. It is a general system that can be used to manage any collection of files. For example, those files might be source code—for others, anything from grocery shopping lists to digital video mixdowns and beyond.

What is a Subversion Repository ?

Subversion is a centralized system for sharing information. At its core is a repository, which is a central store of data. The repository stores information in the form of a filesystem tree—a typical hierarchy of files and directories. Any number of clients connect to the repository, and then read or write to these files. By writing data, a client makes the information available to others; by reading data, the client receives information from others.

So why is this interesting? So far, this sounds like the definition of a typical file server. And indeed, the repository is a kind of file server, but it's not your usual breed. What makes the Subversion repository special is that it remembers every change ever written to it: every change to every file, and even changes to the directory tree itself, such as the addition, deletion, and rearrangement of files and directories.

When a client reads data from the repository, it normally sees only the latest version of the filesystem tree. But the client also has the ability to view previous states of the filesystem. For example, a client can ask historical questions like, "What did this directory contain last Wednesday?" or "Who was the last person to change this file, and what changes did he make?" These are the sorts of questions that are at the heart of any version control system: systems that are designed to record and track changes to data over time.

2.6.1 Features of Subversion

Directory versioning

CVS only tracks the history of individual files, but Subversion implements a "virtual" versioned

filesystem that tracks changes to whole directory trees over time. Files and directories are versioned.

True version history

Since CVS is limited to file versioning, operations such as copies and renames—which might happen to files, but which are really changes to the contents of some containing directory—aren't supported in CVS. Additionally, in CVS you cannot replace a versioned file with some new thing of the same name without the new item inheriting the history of the old—perhaps completely unrelated—file. With Subversion, you can add, delete, copy, and rename both files and directories. And every newly added file begins with a fresh, clean history all its own.

Atomic commits

A collection of modifications either goes into the repository completely, or not at all. This allows developers to construct and commit changes as logical chunks, and prevents problems that can occur when only a portion of a set of changes is successfully sent to the repository.

Versioned metadata

Each file and directory has a set of properties—keys and their values—associated with it. You can create and store any arbitrary key/value pairs you wish. Properties are versioned over time, just like file contents.

Choice of network layers

Subversion has an abstracted notion of repository access, making it easy for people to implement new network mechanisms. Subversion can plug into the Apache HTTP Server as an extension module. This gives Subversion a big advantage in stability and interoperability, and instant access to existing features provided by that server—authentication, authorization, wire compression, and so on. A more lightweight, standalone Subversion server process is also available. This server speaks a custom protocol which can be easily tunnelled over SSH.

Consistent data handling

Subversion expresses file differences using a binary differencing algorithm, which works identically on both text (human-readable) and binary (human-unreadable) files. Both types of files are stored equally compressed in the repository, and differences are transmitted in both directions across the network.

Efficient branching and tagging

The cost of branching and tagging need not be proportional to the project size. Subversion creates branches and tags by simply copying the project, using a mechanism similar to a hard-link. Thus these operations take only a very small, constant amount of time.

Hackability

Subversion has no historical baggage; it is implemented as a collection of shared C libraries with well-defined APIs. This makes Subversion extremely maintainable and usable by other applications and languages.

2.6.2 Basic Operations with Subversion

Creating a Subversion Repository

A repository can be created using the svn tool svnadmin. For example, to create a svn repository named mysvn in /home/myuser/sourcecodes/ directory, open up a terminal, change the Current Directory to /home/myuser and issue the create command of the svnadmin tool, as follows

$ cd /home/myuser/sourcecodes

$ svnadmin create mysvn

This will create mysvn repository in the present working directory which is /home/myuser/sourcecodes

Adding files to the created repository

Now, as we have a repository available, we can add files into the repository to be available under version control. To do this, create a dummy directory structure as required and import them into the newly created subversion repository.

Most subversion repositories of software projects contain three basic directory in the top level named 'branches', 'trunk' and 'tags'. These three are placed in the root directory which is given the name of the project. Taking an example of the project being 'myproject', these are the steps to create a basic directory structure to the repository.

- Change the PWD to home directory,

$ cd /home/myuser

- Create a directory called myproject and change the PWD to it,

$ mkdir ./myproject $ cd myproject

- Create the three basic directories as stated above,

$ mkdir ./trunk $ mkdir ./branches $ mkdir ./tags

- Check whether everything is ok,

$ ls . branches tags trunk

- Change back to the home directory

$ cd /home/myuser

- Import the newly created directory structure to the repository using the import command,

$ svn import ./myproject file:///home/myuser/sourcecodes/mysvn/myproject -m 'first import'

In this statement, there are certain things to note down.

First, while using subversion on the client side, all user commands start with the word 'svn' followed by an option such as 'import', finally followed by arguments and additional options. The typical syntax for a svn user command is of the form

svn option argument [extra_options arguments..]

Thus, the above statement executes the import option. The given argument is the source directory and the address of the destination repository. In our case, the destination repository is in our local file system. It can also be in a remote location or hosted in the Internet.

In the case of a hosted repository whose url is known (ex. http://svn.myhost.com/public/mysvn), the same statement takes the following form,

$ svn import ./myproject https://svn.myhost.com/public/mysvn

The additional option in our case is a '-m', which specifies a message to be saved regarding what is being added. This s helpful when the logs are checked to find what was actually done with the repository.

Now, the basic directory structure has been added to the repository. In addition to just the basic directory structure, we can also have some files within them. These files will be added to the appropriate directory when import statement executes.

These are the basic steps of creating a repository and adding a basic file structure to it. Most probably in real life situations, there will be repository already available or your web host will help you in making a repository for you. Having seen the basic operations to be performed on the server end, let us see the operations which can be performed from the client side which is more important to us.

Checking out a working copy from the repository

What we created was a subversion repository, a central place where our files are going to be stored. This will act as a source for our files. the most basic operation to be done with an existing repository is to get the files present in the repository.

An important point to note at this juncture is that when we fetch a copy of the repository, we are fetching just a **working copy** of it which is independent to the content in the repository. To be more precise, a checked out copy is a local copy of the repository which contains the files which were present under the latest revision or state of the repository when we did a check out. Having a working copy does not guarantee us a permission to add files, update files or delete files from the repository. But, as we have a copy of the files, we have full freedom to play around with it without worrying that we are going to affect the original repository.

Another interesting thing which is hidden within this working copy is that it has some details regarding the revision of the repository it corresponds to as well as details regarding the source repository and our permissions over it. Thus, after some time, if we want to update our working copy to a latest version in the repository, we just need to issue an update command. Viola! All our files get updated to the latest revision, but if we had made some changes in the files either they will be merged with the new updates or will result in a conflict. Lets first check how to check out a working copy before wondering about resolving conflicts.

The repository we created is available at the location */home/myuser/sourcecodes/mysvn/myproject* in our local file system. To get a working copy of it, we have to use the checkout (alias co)command with the name of the working copy we want as the second argument.

$ cd /home/myuser/sourcecodes/

$ svn checkout file:///home/myuser/sourcecodes/mysvn/myproject myprojectWC A branches A tags A trunk Checked out revision 1.

$ ls . mysvn myprojectWC

Thus we have obtained a local working copy named 'myprojectWC' of the repository 'myporject' under the subversion 'mysvn'. The output said what are the files being added into the local copy and finally what is the version currently being checked out.

Always keep an eye on the verbose messages given by svn, they are very informative

What if the repository had been hosted in the web ? This is how we can check out a copy of a repository hosted somewhere,

$ svn co https://svn.myhost.com/public/mysvn/myproject myprojectWC

Note: we used an alias 'co' to the 'checkout' option

Adding files and Committing changes

Now we have a local working copy of the repository, we can try adding files to it and trying to upload them into the repository. Any change which is done on the working copy doesn't affect the repository until we do a 'commit'. It is equivalent to making a commitment that we are consciously modifying something in the repository and we hold the responsibility for the same. For a commit to be made we need a commit permission, which is got in the form of a username and password. We need to specify the username and password only once during the first commit and it is valid for the future commits made from the same working copy.

Let us create a file named 'myfirstfile.txt' and add it to the repository. To do this, create a file named myfirstfile.txt in 'trunk' directory of the repository. Then issue the add command to add it to the next commit and then issue the commit command.

Note: all commands are to be executed from the top level directory of the working copy. In our case, the present working directory PWD needs to be myprojectWC when issuing any svn commands. We can issue from any inner level directories as well, but the changes will be made only to the corresponding directory and within it.

$ cd myprojectWC/trunk

$ cat myfirstfile.txt This is my first file. Am going to add it to my repository. <CTRL-D>

$ cd ..

$ echo $PWD /home/myuser/mysourcecode/myprojectWC

$ svn add trunk/myfirstfile.txt A trunk/myfirstfile.txt

$ svn commit -m 'my firs file addition' Sending trunk/myfirstfile.txt Transmitting file data. Committed revision 2.

Though we have done a commit from the client end, for the revision to be updated in our local records we need to follow every commit command with an update command.

$ svn update At revision 2.

Some points to notice here are, the first revision was numbered 1. When we added file and committed, the revision changed to 2. Likewise, with every change in the repository, let it be addition of files, modification of file content, deletion of files or merging of files, the revision number is moved to the next value.

What actually happens is, instead of adding our new file to the revision 1 contents, it created a new copy of the existing revision, named it as revision 2 and added our new file to the revision 2. Thus, we have our revision 1 intact and we can, if we want to, check out the first revision from the repository. The main advantage of this is,

1. If we want to work with earlier revisions of our repository, we always have it available at a single command.
2. If we ever did a mistake and committed a wrong file or a broken code, we can 'roll-back' to the earlier revision safely. This erases the wrong commit we made to the repository, thus getting us all our precious files intact as in the previous state.

Handling Conflicts

Conflicts occur when we try to apply changes over a revision which is latest compared to the one over which we are working on. That is, the code has changed since we last updated our working copy and hence we need to first update our working copy before we can commit our changes.

A typical example will be that we are trying to commit some changes when our working copy is at revision 89, while the latest revision in the repository is 90. Thus our working copy is outdated with respect to the repository and we might have missed the changes made in revision 90. To solve this we need to update our working copy to version 90, make sure there is no conflict of code with the changes we made and then proceed with a commit.

The conflict is not serious when the file changed and file we are trying to commit are different. The conflict needs to be resolved only when the code which has changed in the latest revision with respect to the last revision and the code we are trying to commit are same. At such a situation we need to look at the code and decide which code needs to remain. That is we have to manually

remove the conflicting part of the code. To solve this conflict we need to issue a 'resolved' command over the file in conflict. Then we can continue with the commit. Hence, it is always advisable to update our working copy before we start working. Conflict which rises due to changes made after we updated has to be resolved manually.

A typical resolve command looks like this,

$ svn resolved trunk/myfirstfile.txt

Removing Files

As files can be added to the repository, files can be also removed from the repository. Removing files from our working copy will not remove the files from the repository, rather it appears back on an update. The only way to remove a file from the repository is to mark it for removal and commit.

For example, to remove the myfirstfile.txt from the repository issue the following commands.

$ svn remove myfirstfile.txt

$ svn commit -m 'file removed'

Status, Info and Logs

There are few commands which will let us know some information about the working copy and the repository. Among them status, info and log are important ones, very useful in day to day operations.

$ svn info

This gives information about the current revision of the working copy and who committed the current revision and when. It also gives the URL of the subversion repository, the root and UID of the repository.

$ svn log

This outputs the entire log of the repository, which gives information about the commits made till now. In the logs, the message added to the commits with the -m option are displayed.

$ svn status

This command gives the current status of the working copy. The main difference of status from info is that info just tells about the version and repository details while status shows the current state of files in the repository.

In the output of status, special symbols are used to mark files for corresponding actions to be taken on them. For example, 'A' marks files to be added to the repository, 'D' marks files to be deleted, 'M' marks modified files and a '?' marks unversioned files.

It is important to do a status check on the working copy before doing a commit, to make sure all the files are marked properly and there is no files left out unversioned. Thus, all the files we need to

commit should not have a '?' mark over them.

Tag

Tags are nothing but simple snaphsots of project in time. Tags are useful in creating snapshots of directories and subdirectories with a human readable name. This is very useful during release periods where a certain revision can be tagged with a release number.

For example, the revision 4330 under the trunk can be tagged as release 2.10. It is easy to be handled with a human readable name like release 2.10 rather than revision 4330 of the trunk. Also, we can tag certain specific sub-directories like i386 sub-directory of trunk and release it as version 2.10-i386, where only the i386 code is released under the version 2.10 while others are not.

To tag the current revision in the trunk as release 1.0, issue the following command

$ svn copy https://svn.myhost.com/public/mysvn/myproject/trunk \
https://svn.myhost.com/public/mysvn/myproject/tags/release-1.0 \ -m 'Tagging release 1.0 of myproject'

Branch

Branch in a Subversion repository is a deviation from the current code in the trunk. A branch is created when doing something different from what is being done in the trunk, but it is still dependent on the code in the trunk.

For example, we are writing a software whose user interface is implemented using GTK tool kit. Now, we also want to support Qt tool kit and hence we create a branch for the same project that is the trunk. In the branch, we implement the Qt based user interface instead of GTK but still the other parts of the software are same.

Branches are generally created when an idea has to be implemented in the project, but not disturbing the development happening in the trunk rather by doing a parallel implementation of the same in the new branch. The advantage of this is, the developments happening in the trunk can still be merged with the branch and hence the branch doesn't lag too far behind the developments in the trunk. The vice versa is also possible, merging code from the branch into the trunk.

Thus, a branch is an independent line of development which doesn't interfere or get interfered by another line, still shares a common history with the other line.

Merging a Branch into Trunk

Often we would have a situation where we need to make the improvements which were done in the branch to be implemented into the trunk. This is done by finding the difference between the trunk and the branch, and merging the difference into the trunk.

The solution is not to create the difference between the current trunk revision and current branch revision and use it to merge the code from branch into trunk. The reason is, when a merge command is given, it actually creates a diff of the two codes in comparison, and in the above case it will generate a diff for not only the additions which were made in the branch, but also the deletions

which happened in the trunk which in no way is connected to branch.

The solution is to find where the branch actually forked off from the trunk and use if as a starting point for the diff, while the ending point is the current revision in the branch, which is known as HEAD revision of the branch. The point of branching can be found by inspecting the logs with the *svn log* command.

For an example, considering that the branch was created at revision 340 and the current HEAD is at 364, all the changes between revisions 340 and 364 in the branch needs to be merged into the trunk. Here is the commands to do it,

```
$ cd trunk
$ svn update
at revision 364

$ svn merge -r 340:HEAD
https://svn.myhost.com/public/mysvn/myproject/branches/june_branch
U db_pgsql.py
U db_utils.py
U urlfetch.py
U urlfetch.tmpl
U browseurl.tmpl

$ svn status
M db_pgsql.py
M db_utils.py
M urlfetch.py
M urlfetch.tmpl
M browseurl.tmpl

$ svn commit -m 'Merging june_branch changes r340:364 into the trunk'
Sending db_pgsql.py
Sending db_utils.py
Sending urlfetch.py
Sending urlfetch.tmpl
Sending browseurl.tmpl
Transmitting file data .....
Committed revision 365
```

There are two important points to note, first is to do a svn update on the trunk before merging anything into the trunk and second is to indicate the revisions being considered for merging in the commit message. The former is important because the merge commands creates a diff in the working copy and then applies it to the trunk in the repository, hence requiring an updated trunk in the working copy as well. The latter is important because, when we continue development in the branch and at a later date we again want to merge the branch, we need not merge it from the beginning when the branch was created. Instead it is enough to merge from where the last merge happened.

The Basic Work Cycle

Although there are lot of commands and options with svn to do a lot of things, there is a general work cycle or sequence of commands to be issued when working with the working copy. This is to ensure that we make good use of the subversion facilities as well as do not waste our time solving problems which could have been prevented by following this work cycle. This work cycle is advised to be followed every time we work with our working copy.

- Update the working copy to the latest revision in the repository, so that we are aware of the changes made in the repository when you start your work.
- Make changes to the working copy, which is what happens during the time of working.
- Make use of the status, diff and revert commands to examine the changes made to the working copy.
- Resolve the conflicts which arise due to changes made by others. This can be done by hand editing, copying file onto the working file in conflict or by reverting the changes we made.
- Commit the changes made to the working copy, with proper commit messages
- Update the working copy to the commit which was just made, by issuing svn update.

2.7 What is Issue Tracking?

As the point needs to be repeated again, to err is human and this holds true to software designed by human as well. No software can be without bugs or more to put it more generic, issues. Issues are something which is not in the right way as it should be. Issues are primarily found during testing before the release, but most issues get detected when the users start using it. Hence it is very vital to get hold of these issues and solve them as soon as possible for the software to attain superior quality.

Since the issues are usually found by the users, it is important to provide a mechanism for the users to report them to the actual developers. Every user who has found an issue in the software he uses really wants it to be informed to the software developer who made the software. But, he might not be willing to undergo cumbersome procedures, especially not mailing the developers and waiting for a reply from them. Rather, what he will like is a simple interface, like a web page, where he can fill the information about the issue in a form and just submit it.

This way of collecting the issues from the user has one advantage. When the issues are sent through mails, the developer may just miss it or forget it and hence the issue goes unattended while the user is disappointed. When he have a proper issue reporting mechanism, it gets registered and the issue is saved in the database. In addition, the issue can be easily tracked and its current status determined. The way to organize the reporting of issue and further management of issue, more precisely tracking of a reported issue is known as Issue Tracking.

2.7.1 The need to report Issues and Bugs

Testing is an important part of the software development process. The main aim of testing is to find bugs and issues which the software may generate during real time use. But still, test conditions are determined by a part of the software development team and may not fully comply with the real time situations. Thus, the software still gets released with bugs and issues, which are eventually found by the user.

In addition to issues being discovered by the user, they may also wish to have certain features in order to improve the usefulness of the software. Thus, the user would like to inform the developers of the issues as well as their feature requests so that they can be taken care of in the next release. An issue tracking system thus gains importance in getting there information from the user and bring into the attention of developers.

An issue tracker also keep a track record of the issues reported and corresponding actions taken. Thus, the developers can find similar issues, track solutions and actions, track who did what and even generate reports on issues that were reported. Therefore an issue tracker is not just a mechanism to report issues but a still higher tool to manage issues and customer support

management.

2.7.2.How the Issue Tracking system works?

The issue tracking system works in a simple way. When the user finds an issue, which may be a bug, an enhancement, a feature request or a support request, he logs into the online web interfaced issue tracking system. This issue tracking system is nothing but a form like interface where the issue reporter has to fill in some information like who is the reporter, his mail id, in which product is he reporting the issue on, which release or revision is it, where is the bug, what is the error message generator or what enhancement or feature he likes to have and why, and many similar information. When this issue is registered, it is saved in a database and given an unique issue number.

On the developer end, the developers are shown a list of newly reported issues. They can view the information, decide whether it is a valid issue or not. If it is not valid, then the issue is closed with a proper reason for it. Else, the issue can be assigned to a particular developer. Then the developer takes the appropriate action to solve the issue. During this process, all actions on the particular issue are recorded in the database so we can always get a history of actions taken on a issue.

Whenever something is edited in the issue, a notice is sent to all involved with the issue. An example is, when multiple issues are duplicates of an issue, then a solution added to the main issue results in the same solution being informed to people associated with the duplicate issues as well. Another feature is that we can search through the existing issues to find whether there already exists a similar issue before reporting one. Thus duplicate issues can be avoided, rather an existing issue can be raised in priority when more people report and add up to the same.

2.7.3 Some Popular Issue Tracking softwares

Trac is an enhanced Open Source issue tracking system, provided with a wiki and various features like version control and report generator. Trac is a Python based application which is widely used as a web based issue tracking solution.

Bugzilla is a very reliable bug tracking software, developed by Mozilla under their software development management toohttp://scarab.tigris.org/ls. It is featured with excellent security, increased performance and scalability and comprehensive permission system.

[Source : http://www.bugzilla.org/]

Scarab is a artifact and issue tracking system developed by Collabnet/Tigris.org and is available under an Apache style license. It is a highly customizable tracking system which includes features such as data entry, queries, reports, notifications to interested parties, collaborative accumulation of comments, dependency tracking.

[Source : http://scarab.tigris.org/]

Mantis Mantis is an easily deployable, web based bugtracker to aid product bug tracking. It requires PHP, MySQL and a web server. It is simpler than Bugzilla and easily editable.

[Source : http://mantisbt.tigris.org/]

2.7.4 Trac

Trac is an enhanced wiki and issue tracking system for software development projects. Trac uses a minimalistic approach to web-based software project management.

Trac integrates a capable wiki engine with a number of project management features. In particular, Trac provides a Web-based interface to a Subversion source code repository, a job/bug ticketing system, and basic scheduling features. Each of these is integrated with the wiki engine. Trac can be readily adapted to nearly any project management style.

The real power of Trac is its ability to adapt itself virtually any project management style. Trac honors wiki formatting across the entire Trac-managed project. Naturally enough, Trac's wiki supports a full range of the usual wiki formatting behavior for links to other wiki pages, lists, emphasis, and so on. Trac, however, also supports special markup for creating links to milestones (milestone:Alpha), tickets (ticket:121), and source code (source:trunk/MyCode.cc). Thus, you can create wiki pages that organize access to any useful selection of project characteristics. Then, to extend the concept, Trac honors wiki formatting not only in wiki pages but in ticket descriptions and even Subversion check-in comments.

Trac allows wiki markup in issue descriptions and commit messages, creating links and seamless references between bugs, tasks, changesets, files and wiki pages. A timeline shows all project events in order, making the acquisition of an overview of the project and tracking progress very easy.

Track Featured Tools

- TracWiki
- TracTimeline -- The timeline provides a historic perspective on a project.
- TracRss -- RSS content syndication in Trac.
- The Version Control Subsystem
    - TracBrowser -- Browsing source code with Trac.
- TracChangeset -- Viewing changes to source code.
- TracRevisionLog? -- Viewing change history
- The Ticket Subsystem
    - TracTickets -- Using the issue tracker.
- TracReports -- Writing and using reports.
- TracQuery -- Executing custom ticket queries.
- TracRoadmap -- The roadmap helps tracking project progress.

The Ticket System

Tracks is loaded with a simple and effective ticketing system to track issues and bugs within a project. The ticketing system is considered to be the central project management element of Trac. Tickets can be issued for for project tasks, bug reports, feature requests and support issues.

The main idea behind this tracking system is to simply the entire process of user interaction and participation in the software development process. Thus, the aim is to make it easy for reporting bugs in the software, asking questions and seeking support, and finally suggesting features which the user wishes to have. Thus, the contributions of the user are recorded and tracked from time to time. This improves the user satisfaction and makes the support system work efficiently.

The ticket system involves creating a ticket regarding a particular issue, which gets assigned to a person who holds the responsibility of resolving the issue. The person can alternatively reassign the ticker to some other. In addition, the tickets can be edited, annotated, assigned, prioritized and discussed at any time.

What does a Ticket contain ?

A ticket contain the following information,

- Reporter - The author who created the ticket.
- Type - The type of the ticket such as enhancement or bug.
- Component - The project subsystem to which this ticket concerns.
- Version - The particular version of the project which is reported.
- Keywords - key words related to the current ticket which will help in organizing tickets and searching them later.
- Priority - Importance of the issue.
- Milestone - A timeline for the issue to be resolved.
- Assigned to owner - Person who holds the principal responsibility.
- CC - A comma separated list of person who needs to be notified about the ticket.
- Resolution - Reason to close the ticket.
- Status - Current status of the ticket.
- Summary - A brief summary about the ticket and the issue being addressed.
- Description - A complete description providing adequate information regarding the issue.

The important advantage in Trac is that these fields can be customized and custom fields can be added. Thus, more information can be sought from the reporter and certain information can be dropped from the default list.

[Source : http://trac.edgewall.org/wiki/TracGuide]

What Happens to a Ticket ?

When a ticket is created, it gets assigned to a person. And, if there are a list of people to be notified it is done. The concerned people check in the ticket and find if it is valid. If not found valid, the ticket is closed with a resolution.

When the issue is found valid, the assigned person can seek additional information from the reporter. Or else he fixes the issue and inform every one about the solution. Else, the assigned person can reassign the ticket to another person.

Whenever some information changes on the ticket it is logged. Thus, the ticket is tracked for each change and this information is available can be viewed as history when a ticket is opened for viewing.

An issue if found a duplicate of another is linked to the original issue and thus the current ticket gets closed having resolved as duplicate .

2.8 Documentation

An important part of any software development process is documentation. Documentation is a process of making a record of information related to the corresponding process being documented. A software documentation involves recording information such as functional and technical requirements, standards, design and implementation procedures, testing process and results, operation procedures, support arrangements related to the software and the data used by the software. In general, a documentation talks about the purpose of the software, requirements of the software, contents of the software, and how to install, configure and use the software.

Documentation is not just a process of recording the information, it also includes the process of making the information available in such a form that the targeted users of the documentation can benefit from it. Documentation is not a process that should be done at the end, before the software is released. It is often overlooked and the importance of documentation in software engineering and development is neglected. Documentation serves as a reference manual for both developers to maintain and improve the software in future as well as for the users on using the software.

2.8.1 Types of Documentation

Design Documentation

Design documentation deals with information about the underlying design details which may not have direct relation with the code of the software but deals with how the software is structured and how its functional components operate, interacting with one another. This document does not describe what the functional component does but what makes the existence of such a functional component in the software. Thus, it specified the role and reasons behind the existence of each and every functional as well as structural components. It can also be said as the lower level documentation rich in explanations and short of technical details.

Design documents are often called as white papers, which makes a comparative study about alternates existing for the current software and justify the necessity of the software. The alternative approaches could be in the data storage layer, logical layer or in the interface layer. It enumerates on the pros and cons of the current implementation and the alternatives available. It also deals without any impartiality, the cost of implementation of the current solution. Thus, a design documentation is not a marketing documentation but which is true in its interpretations about the software's success and it may be completed without making a conclusion.

Technical Documentation

The term documentation is generally interpreted as technical documentation. It is not sufficient to produce a software alone, which only the developer who designed it can understand. Also it is very important to document code for modifications and alterations to be made in future by someone who did not actually write the code. Every piece of code, especially the functional parts of the code, need to be properly accompanied by text which clearly describes what the code does and how. The common practice is to include such documentation text along with the code itself.

This type of documentation, as the name shows, is highly technical in nature and is intended for developers and programmers who need to work with the code. An improperly documented code can be misinterpreted in its functioning and hence, may lead to erroneous modifications of the code. The documentation explains what the variables stand for, what the function does, what are the input arguments, their data types and what is the return value or output of the function.

It has become possible for the developer to auto generate the documentation using the same which was used for writing the code or a different tool. This reduces the load on the developer to write documentation strings as per the observed standards. Also, this helps in keeping the documentation to be upto-date, whenever the functional part of the code is modified. This also helps in preventing the developer to bypass the documentation process as it can be automated without the developer spending his time over it.

The use of technical documentation is limited to those who can actually understand the code and have a necessity to look through the code to make some alterations. Hence, the technical documentations are released along with the source code of the software.

It has been understood that the documentation process is a very difficult one if it is performed after the coding is done and the software is ready to release otherwise. Hence it is insisted that the documentation is always done parallel during the actual time of coding.

User Documentation

User documentation is related to the technical documentation but it is not as much technical. The purpose of the user documentation is to ease the process of learning to use the software, for the user. Therefore, the user documentation concentrates on how to use the software and what are the requirements to install and run the software.

It talks about user centered information such as,

- how to install the software
- what are the tools and features offered by the software
- how to interact with the interface provided by the software
- how to change the settings and configurations to suit user requirements
- How to troubleshoot the problems that might occur while using the software
- How to report issues and problems with using the software
- How to get the support for the software

User documentation requires to be organized in a perfect way such that the user can easily browse through the documentation and get the information that he is looking for. It needs to be properly organized in terms of chapters and indexed. It is also important to reduce the amount of technical jargons and use simple language, such that user understands the documentation with little language capabilities. It is mostly handled by technical writers who specialize in the task of converting technical information into a form which the common man can understand.

Within the user documentation, there are three sub-types each dealing with one type of audience or use.

- **Tutorials** - These documents are targeted over the new user and thus helps him to understand how to use the software, through a guided step-by-step explanation for each and every task.

- **Thematic documents** - These documents concentrate on a specific topic that may be of an immediate interest to the user, like configuring network settings of the software to be able to auto-update the software and auto-download additional plugins from the Internet.

- **Command and Task index** - These documents deals with more technical components like

commands and tasks to be used along with the software. This includes a complete index of all the commands, cross reference with other related articles.

The important point to note is, a single document of one of the above types in not enough. It is advisable to use a combination of all the three types to give a detailed and easy browsable documentation for the user. It is also a common practice to give a tutorial based documentation for the user and offering additional information about commands and tasks through an online documentation. Thus, once the user achieves a comfortable level of using the software with the help of the tutorial, he can browse the online documentation to know more and improve his skills on using the software.

Marketing Documentation

Marketing documentation is intended for promotional activities, to induce an interest in casual observers of the software to try the install and try the software. This necessitates the marketing documentation to be simple yet attractive, initiating an interest to know more about the software. It also needs to clearly explain what the software can do and how it does things better than what other softwares do. It needs more comparative information with other alternatives and the softwares advantages. It is also important to specify the interoperability details of the software with other softwares and devices.

2.8.2.Commonly Used Document Generators

DOxygen

DOxygen is a documentation generator for C++, C, Java, Objective-C, Python, IDL (Corba and Microsoft flavors) and to some extent PHP, C#, D and ActionScript?. It runs on most Unix systems as well as on Windows and Mac OS X. Most of the Doxygen code was written by Dimitri van Heesch. KDE uses Doxygen for parts of its documentation and KDevelop has built-in support for it.

It can generate an on-line documentation browser (in HTML) and/or an off-line reference manual (in $\mbox{\LaTeX}$) from a set of documented source files. The documentation is extracted directly from the sources, which makes it much easier to keep the documentation consistent with the source code. It supports the documentation tags used in the Qt toolkit and can generate output in HTML as well as in CHM, RTF, PDF, LaTeX, PostScript? or man pages

We can configure doxygen to extract the code structure from undocumented source files. This is very useful to quickly find your way in large source distributions. We can also visualize the relations between the various elements by means of include dependency graphs, inheritance diagrams, and collaboration diagrams, which are all generated automatically.

[Source : http://www.stack.nl/~dimitri/doxygen/]

Epydoc

Epydoc is a documentation generator that renders its own lightweight markup language Epytext for Python documentation strings. As opposed to freeform Python docstrings, reStructured Text (both also supported) and other markup languages for docstrings, Epytext supports linking between different pieces of documentation.

There are several tools for rendering Epytext. Most commonly, the epydoc program is used to render Epytext as a series of HTML documents for display on the Internet, or as a PDF document for printing. Epydoc also supports viewing the rendered documentation within Python using a GUI. The syntax is uncomplicated enough for the programmer to read the raw Epytext docstrings embedded in the source code directly.

KDOC

KDOC is a C++ and IDL interface documentation tool, initially written for the specific purpose of generating documentation for the KDE libraries. It extracts specially formatted documentation and information about your classes from the class' header or IDL files, and generates cross-referenced HTML, LaTeX or Man pages from it.

KDOC allows groups of classes to be formed into "libraries" and documentation from separate libraries can be very easily cross-referenced. KDOC offers the following features,

- Generates HTML, Man-page and LaTeX output.
- Cross-reference classes in separate libraries.
- Supports Qt signals and slots.
- Written in perl and easily extensible.

When you process a set of C++ headers with KDOC it creates

- A set of HTML files (two index files and one for each class) and/or
- A set of man pages (one index page and one for each class) and
- A LaTeX file (containing all docs for the classes) and
- A file with a .kdoc extension, which can be used to reference classes and members of this library from other libraries.

[Source : http://www.ph.unimelb.edu.au/~ssk/kde/kdoc/ http://sirtaj.net/projects/kdoc/]

phpDocumentor

phpDocumentor, sometimes referred to as phpdoc or phpdocu, is the current standard auto-documentation tool for the php language. Written in php, phpDocumentor can be used from the command line or a web interface to create professional documentation from php source code. phpDocumentor has support for linking between documentation, incorporating user level documents like tutorials and creation of highlighted source code with cross referencing to php general documentation.

The important features of phpDocumentator are,

- output in HTML, PDF (directly), CHM (with windows help compiler), XML DocBook?
- very fast
- web and command-line interface
- fully customizable output with Smarty-based templates
- recognizes JavaDoc-style documentation with special tags customized for PHP 4
- automatic linking, class inheritance diagrams and intelligent override
- customizable source code highlighting, with phpxref-style cross-referencing
- parses standard README/CHANGELOG/INSTALL/FAQ files and includes them directly

in documentation
- linking between external manual and API documentation is possible at the sub-section level in all output formats
- easily extended for specific documentation needs with Converter

[Source : http://phpdoc.org/]

2.9 LaTeX - A Document Preparation System

LaTeX is a high-quality typesetting system; it includes features designed for the production of technical and scientific documentation. LaTeX is the de facto standard for the communication and publication of scientific documents. LaTeX is based on Donald E. Knuth's TeX typesetting language or certain extensions. LaTeX was first developed in 1985 by Leslie Lamport, and is now being maintained and developed by the LaTeX3 Project.

LaTeX is a document preparation system for high-quality typesetting. It is most often used for medium-to-large technical or scientific documents but it can be used for almost any form of publishing. LaTeX is not a word processor. Instead, LaTeX encourages authors not to worry too much about the appearance of their documents but to concentrate on getting the right content.

2.9.1 Features of LaTeX

The important Features of LaTeX are,

- Typesetting journal articles, technical reports, books, and slide presentations.
- Control over large documents containing sectioning, cross-references, tables and figures.
- Typesetting of complex mathematical formulas.
- Advanced typesetting of mathematics with AMS-LaTeX.
- Automatic generation of bibliographies and indexes.
- Multi-lingual typesetting.
- Inclusion of artwork, and process or spot colour.
- Using PostScript? or Metafont fonts.

=== Example of a LaTeX Document ==

This is a small example of how a typical LaTeX file may look like,

hello.tex

```
\documentclass{article}
\title{Cartesian closed categories and the price of eggs}
\author{Jane Doe}
\date{September 1994}
\begin{document}
   \maketitle
   Hello world!
\end{document}
```

2.9.2. LaTeX Commands

LaTeX commands start with a backslash \ and is followed by a name consisting only of letters. The characters like # $ % ^ & _ { } ~ have special meaning in LaTex? and hence need to be escaped with a \ when used in the text. Some LaTeX commands consist of a backslash and exactly one letter. White-spaces after the command are generally ignored. The command is then followed by an option within [] and/or an argument within {}.

2.9.3 Basic LaTeX File Structure

Every input file must start with a *documentclass* command, which specified the class of the type of document we intend to write.

\documentclass[options]{class}

The class command can be followed by other commands which alter the style of the documents We can also include packages that add new features to the document. To load a package use the *usepakcage* command,

\usepackage[options]{package}

When the style setup is finished, we can begin the actual document using the *begin* command. The area between the class and begin command is known as *preamble*.

\begin{document}

A document which was begun by the above command should be, at the end of the document, be ended with an *end* command.

\end{document}

Anything that is entered after the end command will be ignored by LaTeX.

2.9.4 Document Class

Document class defines the type of document that the author wants to create. This is specified with a *documentclass* command, where the *class* specifies the type of the document. The 'options' parameter customizes the behavior of the class. Multiple options can be given for a class as a comma separated list. The common document classes are

- article for scientific journals, reports, presentations, etc.
- minimal only sets page size and font
- report for longer reports containing many chapters
- books for real books
- slides for presentation slides

Some common class options are,

- 10pt, 11pt, 12pt for size of the main font.
- a4paper, letterpaper... specifies the paper type.

- titlepage, notitlepage specifies whether a new page should be started after the document title or not.
- onecolumn, twocolumn for column type.
- twoside, oneside for double or single sided output.
- landscape changes the layout to landsacpe.
- doc allows documentation of LaTeX programs.

2.9.5 Packages

When we actually create a document, there are some common problems which can not be handled by LaTeX. For example, inclusion of images, graphs, colored text or source code inclusion into a file. This requires to enhance the capabilities of LaTeX such that these tasks could be done, which needs the help of packages. This is done by loading up additional packages using the *usepackage* command.

2.9.6 Page Style

Pagestyles are nothing but a specific combination of header/footer and LaTeX supports 3 page styles. It is defined with the command,

\pagestyle{style}

where style can be one of the following.

plain prints the page numbers on the bottom of the page, in the middle of the footer. This is the default style.

headings prints the current chapter heading and the page number in the header of each page, while the footer remains empty.

empty makes the header and footer to be empty.

Including Files

As in programming languages, LaTeX allows the inclusion of contents from other files into the body of a LaTeX file. This is done with the help of *include* command.

\include{filename}

When the file is inserted, LaTeX starts a new page from the beginning of the included text. Inclusion can thus be made at any part of an existing file. Thus, instead of making one big file, it can be divided into smaller files and included in a sequence.

2.9.7 Running a LaTeX File

To run a LaTeX file, a working installation of LaTeX is needed. This is mostly available in the standard packages of most Linux distributions and hence can be installed using the corresponding package managers.

A LaTeX file is a plain ASCII text file, saved with an extension *.tex*. When LaTeX is run on this file, it will create a *.dvi* file upon successful completion. If there is any error in the commands use, the process will stop where the error occurred, the error being displayed in the terminal where from the file was run.

Create a file called as hello.tex using the example shown before. Then run this command in a terminal(or console).

$ latex hello.tex

This produces a DVI file called hello.dvi, which can be viewed with a DVI viewer such as xdvi or kile. Alternatively, the .dvi file can be converted to PDF or PostScript? form.

In this chapter, we have studied the special features related to the collaborative nature of development in Free/Open Source Software. The structure and dynamics of community and how they interact using mailing lists, wiki, chat and messaging are described. The features of integrated development environment are discussed with Eclipse as an example. Version Control Systems, with Subversion as an example are described. The need for Issue Tracking is brought out and how the system functions is described with the example of Trac. Documentation in FOSS is described using LaTex example.

Chapter 3

Presentation Layer

In many computer applications where lots of data are presented in a multi-user environment, it is preferable to separate the data modeling aspects from the user interface aspects. This will ensure that the changes to the client environment, depicted through the user interface, do not affect the data handling. Further, the data can be reorganized independent of the user interface concerns. This is achieved in the Model-View-Controller (MVC) paradigm, discussed in a later chapter. The current discussion covers the aspects of user interface in the three layered architecture, focusing on the ***View*** part of the MVC paradigm. The presentation layer provides the user interface for the application. It provides facilities for **input**, allowing the user to control the application and **output** which provides the results to the user. As the computer applications cater to different types of clients using desktops, workstations and mobile devices, many types of user interfaces can be used in an enterprise application. After a brief discussion on the various client environments and user-interface types, the present chapter deals with the use of packages like HTML, XHTML, etc, for the presentation of the contents and studies the example of CSS (Cascading Style Sheets) for presentation of the style. This prepares the background for discussing dynamic management of web contents using Javascript, DOM and Ajax discussed in the last three sections..

3.1 Client Environment.

In a multi-user environment, the client systems will have different hardware and software capabilities. Often, a client could be a desktop or workstation running different client-server applications accessing a varied set of servers. One could even find a scenario where a given system will be a server for one application while being a client for another application. The clients generally run a GUI or web browser based user-interface applications to process data and access the server. Most of the servers are web based and hence the web browsers have become the most common means of accessing the servers, whether on the web or on the local network.

Apart from the usual desktops, mobile devices and thin client systems are also getting widely used as clients. While the increased penetration of web in the communication applications demands enhanced mobile applications, the desire to minimize the cost and client-side load brings the focus on thin client systems.

Mobile Devices:

Mobile devices like personal digital assistants (PDAs), cell phones, hand held game consoles are also now gaining the ability to access the web. In these systems, the input and output are generally combined into a touch-screen interface. The mobile devices are generally associated with the following functionalities :

- limited data exchanges through small, primarily text-based screens, e.g., for SMS and WAP
- basic data access through menu or icon-based navigation for e-mail, address book, SMS and a basic web browser
- enhanced data access for office applications and custom corporate applications, over and above the basic data access

However, most of the mobile clients are still web browsers which support a subset of the features available on the desktop versions. They also lack the plug-in functionality of the bigger versions. We will take a brief look at working with this type of clients also.

Thin clients:

Often, client-server architecture networks use the central server heavily for processing activities while the client systems are mostly dumb terminals without hard disk drives, used primarily for input / display functions. Such thin or lean clients will require a small boot image essentially to connect to a network and start up a dedicated web browser or remote desktop connection. In contrast, the “thick or fat clients” do most of the processing locally and pass only the data required for communications or archival storage to the server.

While designing a client-server application, the cost, robustness and security considerations have a bearing on allocating the tasks to be handled by the server and the clients. Flexibility of the design for later modifications, enhancements and porting also can influence the design. One can minimize the cost on developing custom clients by using a standardized client software such as a web browser. A great deal of client software is typically included in a base boot image, specifically to support various user applications, so that it need not be reinstalled on every client computer. Typically, the user will have only a screen, keyboard, a pointing device (if needed) and enough computing power to handle display and communications.

The advantages of the thin clients include the following :

- lower IT administration costs since they are mostly managed at the server
- lower hardware costs since the clients need minimal hardware
- easy to secure application data which resides only on the server
- low energy consumption by the clients and could even be used in dusty environments
- less network bandwidth since most of the traffic consist of mouse movements, keystrokes

and screen updates
- efficient use of resource and easier to upgrade

However, the clients cannot have local variations and are expected to have the replication of OS and other client software. They are also considered to be not well suited for multimedia-rich applications.

3.2 User interface types

The most common user interface types for applications are:

1. Graphical User Interfaces (GUI) using the Operating system's native drawing capabilities to draw windows and other controls on the computer monitor to interact with users.
2. Web Based Interfaces which produce web pages which are transmitted over a network and then displayed to the user by a web browser.

Of these two types, the web based interface can be considered as a subset of the general graphical user interface. However there are enough differences between the two to have them as distinct types.

Some other types of user interfaces that can be used are:

1. Command line interfaces where the user interacts with the application by entering commands in a special command line.
2. Touchscreen interfaces using a touch sensitive screen for getting inputs
3. Voice activated interfaces where the user can interact with the application with voice commands.

GUI applications are usually written using the native API of an operating system and are often specific to the OS. They have to be installed in the user's system and run locally. There is no standard way of deploying applications across machines with different operating systems. On the other hand, they are responsive to user input and match the look and feel of the host operating system, thereby providing a *user-friendly* environment.

Web based user interfaces are web pages produced by the application which are then used for interacting with the user. They need not be installed on the user's system and can be run off a web server. There is no special deployment required and updates are easy as updating the copy on the server ensures that users get the latest version. They can also be fully cross platform. The user's OS or CPU architecture does not matter as long as there is a web browser available for the platform. Web based interfaces also have the advantage of supporting more than just desktops. Properly designed applications can be accessed using mobile phones, screen readers etc. On the flip side the responsiveness of the application is dependent on the network connection between the server and the users computer. The available GUI controls are also not as rich as native applications.

The remainder of this book will focus on constructing enterprise applications which utilize a web based interface. This is mainly due to the relative ease with which such applications can be built and tested, and the wide variety of platforms that can access the application.

Web Browser

In an enterprise application, a web browser is a software application which is the main means of accessing the user interface. It enables a user to display and interact with text, image or other type of information in a web page. The web page itself might be located in a remote web site or in a local area network. There may be hyperlinks in the web page pointing out to other web pages. Internet Explorer, Mozilla Firefox, Safari, Netscape, etc. are the examples of web browsers available for personal computers. Though the browsers are primarily used to access the World Wide Web, they can also link to information provided by private network web servers or content in file systems. A brief look into the architecture of a web browser follows.

A web browser connects to a web server through the http protocol and renders the web pages it receives on the screen. In addition to rendering html present in web pages, it also supports many image formats and can support wider variety of formats using other helper applications called plug ins.

Plug ins (or plug-ins) are small pieces of software that interact with the browser to provide certain, usually very specific, functions. Typical examples are plug ins to display specific graphic formats, or to play multimedia files. Some common plug ins include Macromedia Flash, Adobe Acrobat, Apple Quicktime etc.

Cookies

When web browsers connect to a server, the server initially sends parcels of text known as HTTP cookies, also knows as just cookies, to the browsers which are in turn, sent back unchanged by the browser each time it access that server. The term cookie, in Unix parlance, refers to a token or a short pocket of data passed between communicating programs. A cookie is not a computer program, but is simply a bundle of data which cannot perform any operation by themselves. The cookies are used as ticket for authentication, tracking and for maintaining specific information about users. Due to Internet privacy concerns related to cookies, most modern browsers allow users to decide whether to accept cookies. However, rejection of cookies might mean that the user cannot access certain websites. Cookies generally provide a virtual “shopping basket” and are often used for personalization based on user preferences.

The cookie is an arbitrary packet of data chosen by the web server and sent to be browser. By returning a cookie to the web server, the browser provides a means of connecting the current page view with the prior views of the page. Otherwise, each retrieval of a web page or its components will be considered as an isolated event. Cookies can also be set by script in a suitable scripting language, provided this process is supported and enabled by the web browser. The browsers are expected to support a minimal number of cookies or amount of memory for storing them. Typically, an Internet browser can store at least 300 cookies or 4 Kb size or at least 20 cookies per server or domain. The cookie names are case insensitive. In case a computer has multiple browsers, each browser has a separate storage area for cookies. Thus, the cookies refer to a combination of a user account, a computer and a web browser.

Some browsers provide features by which the functionality of the browser itself can be modified to a limited extent. They are called **extensions**. Extensions help the browser to be a small download while any required functionality can be added using extensions. The Firefox browser provides many extensions which can be installed to customize the browser to the user's specific need.

The browsers also might contain **applets** which are software components that run in the context of the browser program. They are used for very narrow functionalities which generally have no independent usage. They execute only on the "client" platform environment of a system as contrasted with "servlets" which add dynamic content. An applet is written in a language, e.g., Java or Flash, different from the scripting or HTML language that invokes it. It usually features display and graphics, often providing an interface for the human user. It must run in a "container" which is provided by the host program, through a plug in or other applications that support the programming model.

During the ensuing discussions, all the concepts and exercises in this section will be based on the Mozilla Firefox Browser version 1.5 as the example. The developed user interface can be tested using a variety of browsers across different platforms. Any exceptions to this rule will be clearly specified.

Web page design

For web page design and client-side scripting, a variety of tools and packages are available. The list includes HTML, CSS, XHTML, Mobile XHTML, XML, Xpath, XSL, DOM, Javascript, AJAX, etc. For authoring the web pages any text editor will do, though an editor developed with web development in mind will ease the development. For instance, one can use Quanta Plus Editor which is a part of the Web Development module of KDE. Finally, the reader must understand that in an actual application the coding for the user interface will not be done by hand. Special middle ware will generate all the web pages for use. Due to this reason, browser incompatibilities are not covered in detail. Use of middle ware will be covered in later chapters. This section is thus mainly for understanding what happens behind the scenes.

We will initially focus on creating static web pages, which were the earliest types of web pages. A static web page means that once the browser receives the page from the server, there is no change in the page's contents until a new page is sent from the server again. Static pages use only HTML and optionally CSS for formatting. The next two sections will cover HTML and CSS, while the later sections in this chapter will give an overview of XHTML, DOM and other technologies.

3.3 HTML

HTML stands for Hypertext Markup Language and is used to design standard compliant web pages which can be viewed in web browsers. HTML uses markup tags to structure information – like classifying some text as headings, paragraphs etc., to describe the structure and meaning of the document.

HTML was developed by Sir Tim Berners-Lee while he was a researcher at CERN. It was then further developed by the Internet Engineering Task Force (IETF). Now the HTML standards are maintained by the **World Wide Web Consortium** (W3C). Many versions of HTML were developed over time. The version 4.01 of HTML is supported by almost all available browsers. It is unlikely that there will be a newer version of HTML as W3C now has XHML as a successor to HTML. In this book unless otherwise specified, HTML means W3C's HTML 4.01.

A HTML document consists of:

1. The Document Type Definition (DTD) which specifies what version of HTML the document follows.
2. Entities which are the various HTML tags used to describe the page. There are different types of markup tags like
    - Structural markup which describes the structure of the document like headings, paragraphs etc.
    - Presentational markup which describes how the content is to be shown, e.g., bold, italic etc.
    - Hypertext markup links which permits linking of various parts of the document or other documents
3. Attributes of the various entities which define some predefined properties of the entities.

The various categories of display elements are :

- Base HTML tags – deal with the structure of the document
- Text – define text properties
- Lists – for displaying information in a list
- Links – hyper links to other documents/sections
- Images – displaying images in web pages
- Tables – displaying tabular data
- Forms – taking input from users
- Frames – dividing the document into frames
- Scripts – integrating some programming within web pages

Base HTML Tags

These tags define the structure of the document. Most of them are mandatory and hence should appear inside each HTML document.

<!DOCTYPE>

A valid HTML document declares what version of HTML is used in the document and names the document type definition (DTD) used in the document. The DOCTYPE tag is found at the beginning of the document. There are 3 valid DTDs available for use with HTML.

- HTML 4.01 Strict DTD – includes all elements and attributes that have not been deprecated or do not appear in frameset documents.
  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
- HTML 4.01 Transitional DTD – includes everything in the strict DTD plus deprecated elements and attributes.
  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
- HTML 4.01 Frameset DTD – includes everything in the transitional DTD plus frames as well.
  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">

<html>
The <html> tags marks the beginning of HTML content in the document. The </html> tag marks the ending of HTML content.

<head>
The HEAD element contains information about the current document, such as its title, keywords that may be useful for search engines, and other data that is not considered document content. The </head> tag closes the head section.

The <title> element provides a title or a heading of the document. It can be present only in the <head> section of the document. Any text enclosed within the <title></title> elements is displayed in the browser's title bar.

The <meta> tags allow authors to specify meta-data – information about the document rather than document content – in a variety of ways.

<META name=”Author” content=”Author Name”> -- others include copywrite, date, keywords.

<body>
The BODY of the document contains the document's content. The content may be presented by a user agent in a variety of ways. The </body> tag closes the body section. Certain visual document properties can be specified within the <body> tag, such as : background, text and link

<body background=”white” text=”#afafaf” link=”green”>
[... content ...]
</body>

A sample html coded document is given below:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title> My first HTML document </title>
<meta name="author" content="S.Senthil Anand">
<meta name="keywords" content="Linux, HTML">
</head>
<body background="white" text="#afafaf" link="green">
Hello World!
</body>
</html>
```

The output of the html page when opened using Mozilla Firefox will be as shown in Fig. 3.1

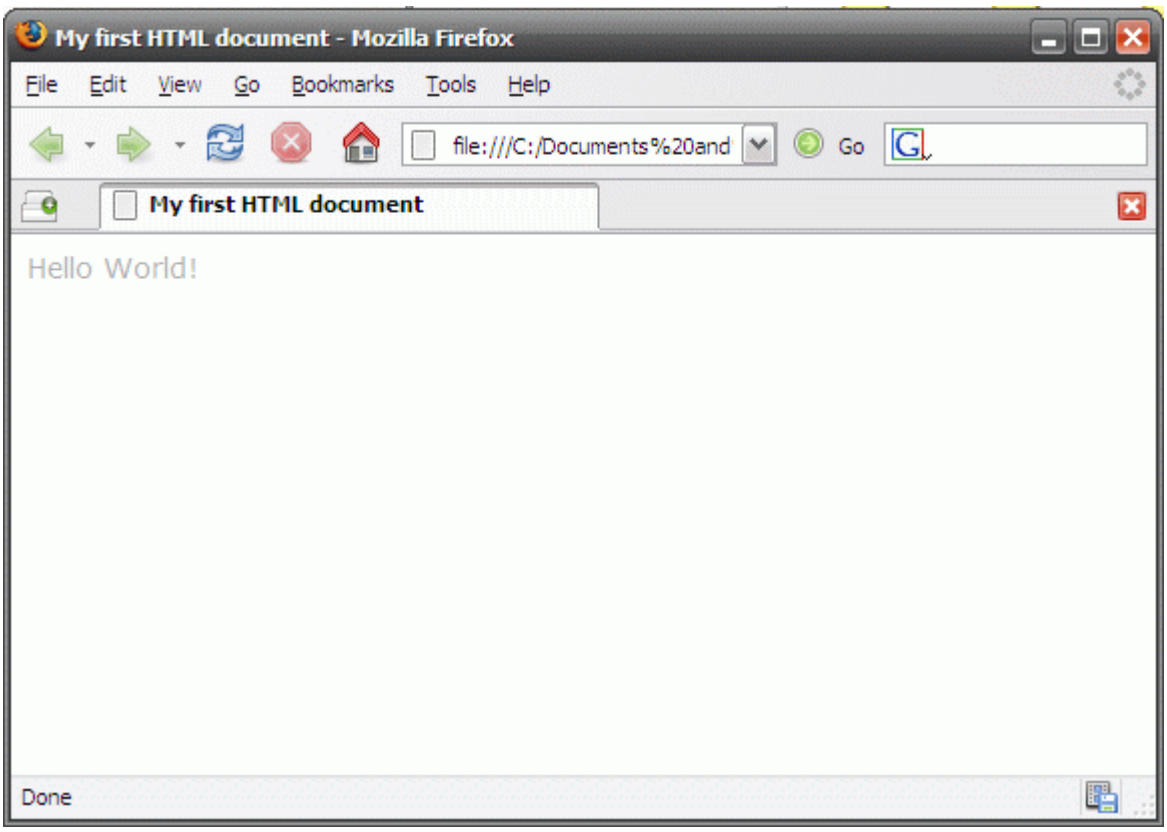

Fig. 3.1 Sample HTML output using base tags

Text Tags

Whitespace has no meaning in HTML. A HTML parser will ignore all whitespaces. Non breaking whitespace can be inserted using the ** ** tag.

 -- line breaks
<p> -- paragraph breaks
 -- emphasis (usually displayed as italics)
 -- stronger emphasis (displayed as bold text)
<code> -- designates a fragment of computer, usually displayed with a monospace font.
<abbr> -- indicates an abbreviated term
<blockquote> -- used to define quotations in HTML text, usually indented by the browser
<sub> -- subscripts
<sup> -- superscripts
<pre> -- preformatted text which is displayed as it is

A sample output generated using these tags in a document is presented in Fig. 3.2:

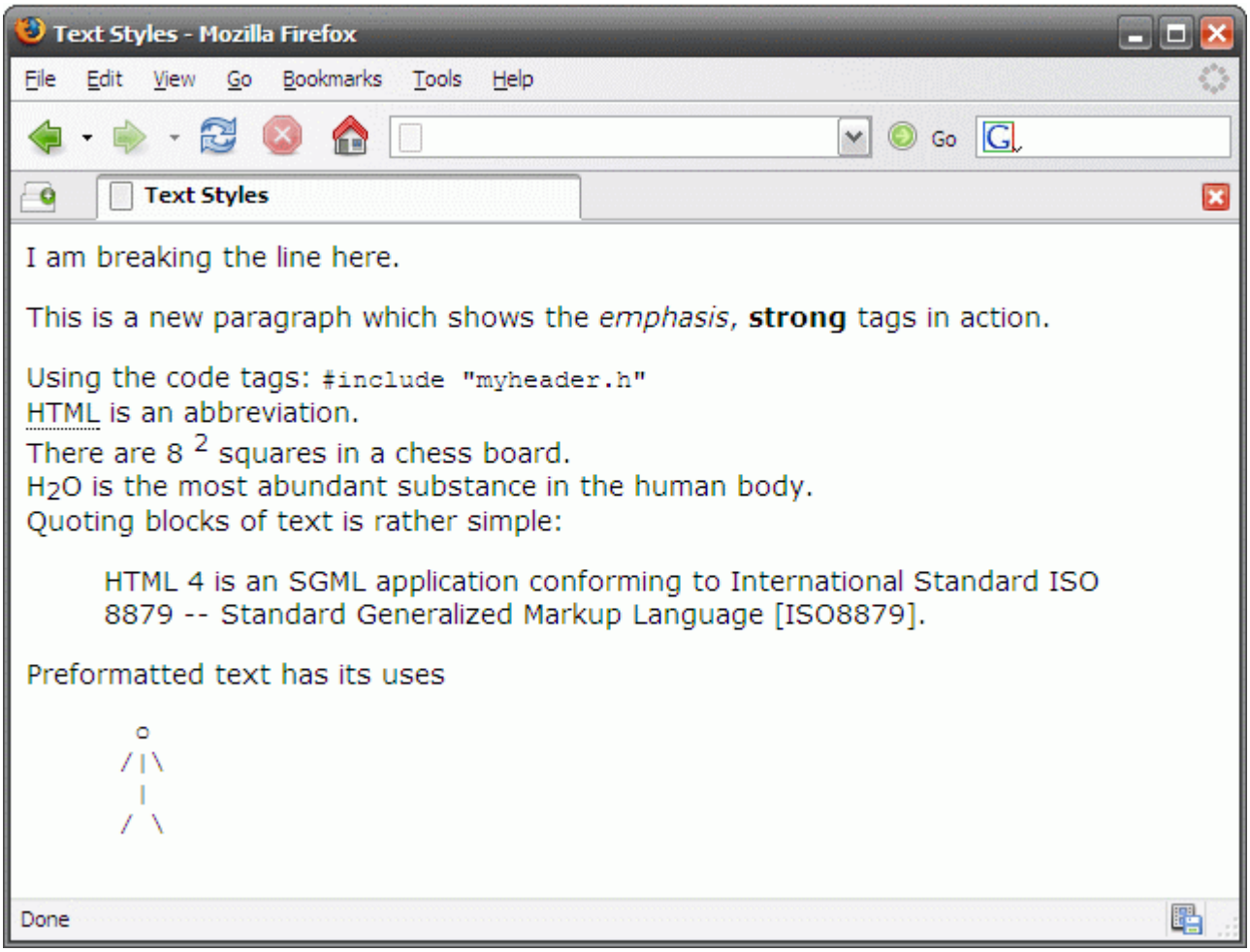

Fig. 3.2 Sample HTML output using Text tags

Headings

HTML allows for multiple heading tags to enable varying size text typically used for headings. Using heading tags also allows a consistent look and feel to be maintained throughout the document. The tags are <h1>,<h2>...etc. The font size decreases as we go.

Lists

- Unordered Lists <ul> -- rendered as bulleted lists
- Ordered Lists <ol> -- rendered as lists with numbers or alphabets and is used where sequence of items need to be emphasized.
- Definition Lists <dl> -- used to make lists for defining items.

A sample html code using List tags is given below.

```
<h3>Unordered Lists</h3>
<ul>
      <li>Keyboard
      <li>Mouse
```

```
        <li>Touchpad
</ul>
<h3>Ordered Lists</h3>
<ol>
        <li>Write Program
        <li>Compile Program
        <li>Run Program
</ol>
<h3>Definition Lists</h3>
Consists of two parts, the term and the definition
<dl>
        <dt>Mouse
        <dd>A small rodent
        <dt>Mouse
        <dd>A computer input device
</dl>
```

The result of the above code in Mozilla Firefox is presented in Fig. 3.3.

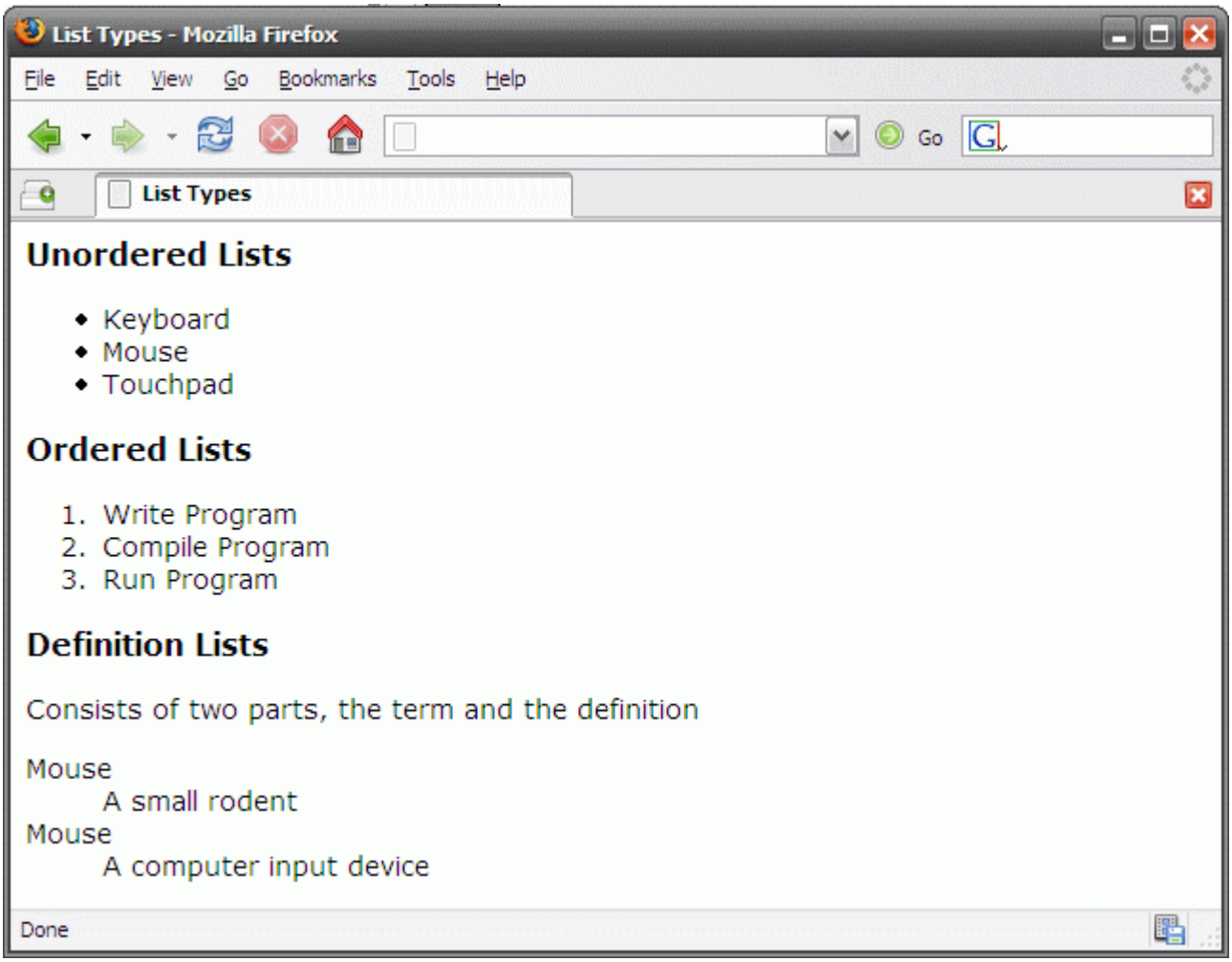

Fig. 3.3 Sample HTML output using List tags

Links

Hyperlinking is an important part of the WWW. Hyperlinking lets the users to provide a link to

other documents as well as sections within the same document. The <a> tag is used to define links as well as anchors. Anchors lets one define placeholders for text. This allows users to jump straight to specific portions of a document.

Format : <a name="ANCHOR" href="URI" title="TITLE">

```
<a href="http://www.gnu.org" title="Click to visit"> The GNU Project </a>
<a name="section-1"> Section-1</a>
<a href="chap3.html#section-1"> Ch3 Sec 1</a>
```

Images

Images can be included into a document using the <img> tag

Format: <img src="IMAGEURL" alt="Alternate Text" height="HEIGHT" width="WIDTH" title="TITLE" align="top|middle|bottom">

where

- src specifies the URL for the image. It can be absolute or relative.
- alt lets us specify a line of text to display if its not possible for the browser to display the image. (eg. in text based browsers or with normal browsers with images turned off).
- The height and width let us alter the size of the image. Specified in pixels.
- title, like for links, allows us to define a pop-up title for the image.
- align specifies the relative alignment of the image with respect to the text.

Tables

Tables are used to display content in a tabular format.

Base -- <table> </table>

Headers and Footers -- <thead> </thead>, <th> </th>, <tfoot> </tfoot>. These tags can be used to designate and format certain content as the header or footer of the table.

Table body, rows and cells -- <tbody> </tbody>, <tr> </tr>, <td> </td>. The rows and columns in a table are defined implicitly based on the number of <td> </td> tag pairs inside each row of the table.

The <table> tag has a number of attributes.

```
<table>
        align="left | right | center"
        cellspacing="NUMBER"
        cellpadding="NUMBER"
        border="NUMBER"
        bgcolor="COLOR"
        rules="ROWS | COLS"
        width="NUMBER | PERCENT" >
</table>
```

A Table Example given below will result in the output as given in Fig. 3.4.

```
<table name="tab1" cellspacing="0" cellpadding="1" border="1" align="center"
width="75%">
<thead>
<tr> <th> Column1 </th> <th> Column 2 </th> </tr>
</thead>
<tbody>
<tr> <td> cell-1,1 </td> <td> Cell-1,2 </td> </tr>
<tr> <td> cell-2,1 </td> <td> Cell-2,2 </td> </tr>
</tbody>
</table>
```

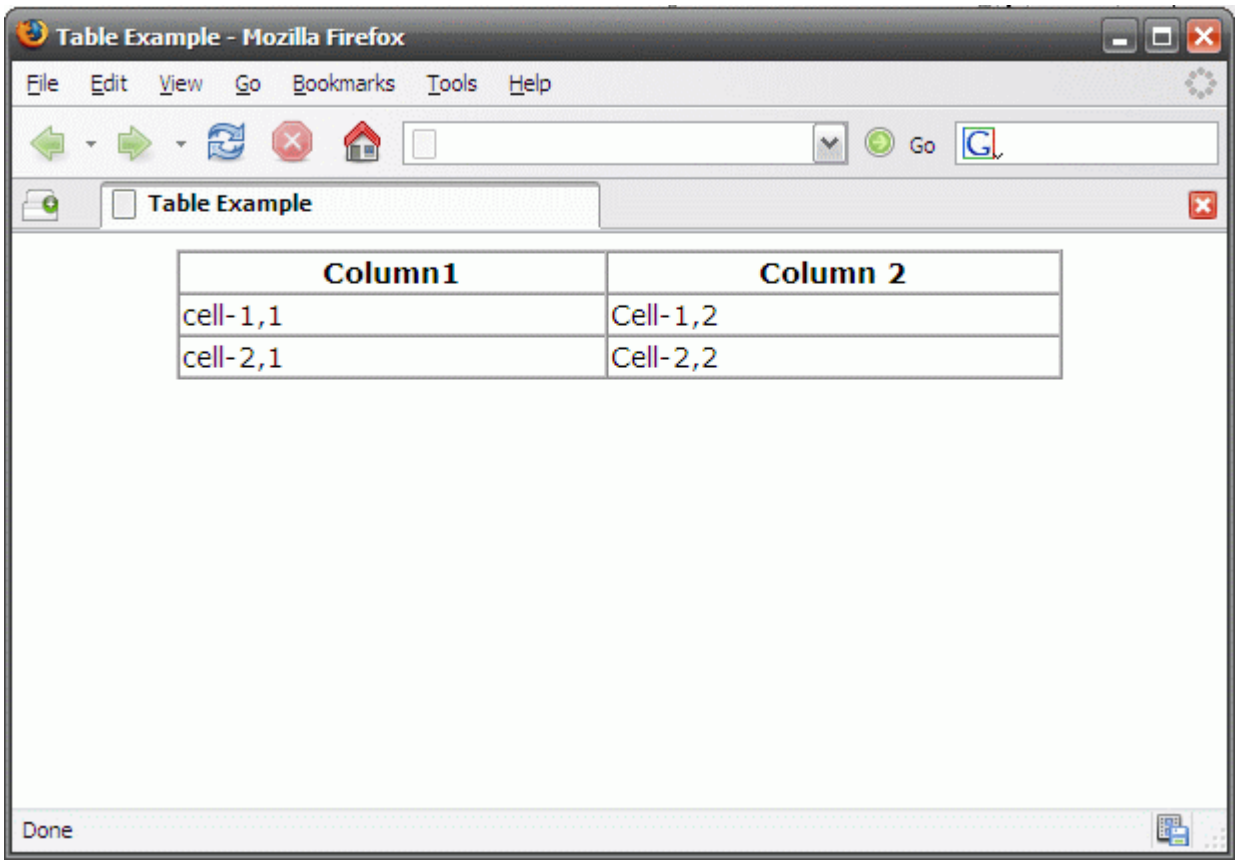

Fig. 3.4 Sample HTML output using Table tags

Forms

Forms are used to take input from the user. Forms can be used to ease and thereby increase the interactivity of a web site or web application. They can contain a number of user interface elements like text boxes, password boxes, check boxes, radio buttons, buttons, text areas, combo boxes etc.

The <form> element has the following attributes:

action – server side form handler
method -- (GET | POST)
name – name of form for scripting

Input to the form is done using the <input> element which has the following attributes:

type – what type of widget is needed
name – submit as part of form
value – specify for radio buttons and check boxes
checked – for radio buttons and check boxes
disabled – whether the element is usable or not
readonly – no changes can be made
size – specific to each type of field
maxlength – max. characters for text fields
tabindex – position in tabbing order

The various types of input widgets are :

text: Creates a single-line text input control.
password Like "text", but the input text is rendered in such a way as to hide the characters (e.g., a series of asterisks).
checkbox: Creates a checkbox
radio: Creates a radio button
submit: Creates a submit button
reset: Creates a reset button
button: Creates a generic push-button
hidden: Non-visual element that can be used for passing information between the browser and the web server
file: Creates a control to upload a file to the server
image: Creates a graphical submit button

The <select> element allows users to select options.

name – field name
size – rows visible
multiple – default is single selection
disabled – user is not allowed to interact with element

The <option> element can occur within a <select> </select> tag

selected -- This option is selected by default
disabled -- This option is not selectable
value -- This is the value that is sent if this option is enabled.

Textarea is used to get multiple lines of text as input.

name -- name of the textarea
rows -- number of rows to show
cols -- number of columns to show
readonly -- can the content be changed?

A forms example given below will yield an output given in Fig. 3.5.

```
<form action="/cgi-bin/process_form.cgi" method="post" name="AddUser">
Name: <input type="text" name="name" size="20" maxlenght="30">
Sex: <select name="sex">
<option value="male"> Male </option>
<option value="female"> Female </option>
</select>
<br>
User on this system?: <input type="checkbox" name="user" value="yes"> Yes <br>
Allows to access application?:
<input type="radio" name="accessApp" value="yes">
<input type="radio" name="accessApp" value="no">
<br>
Other details:
<textarea rows="1" cols="30"></textarea>
<br>
<input type="submit">
<input type="reset" value="Erase Data">
</form>
```

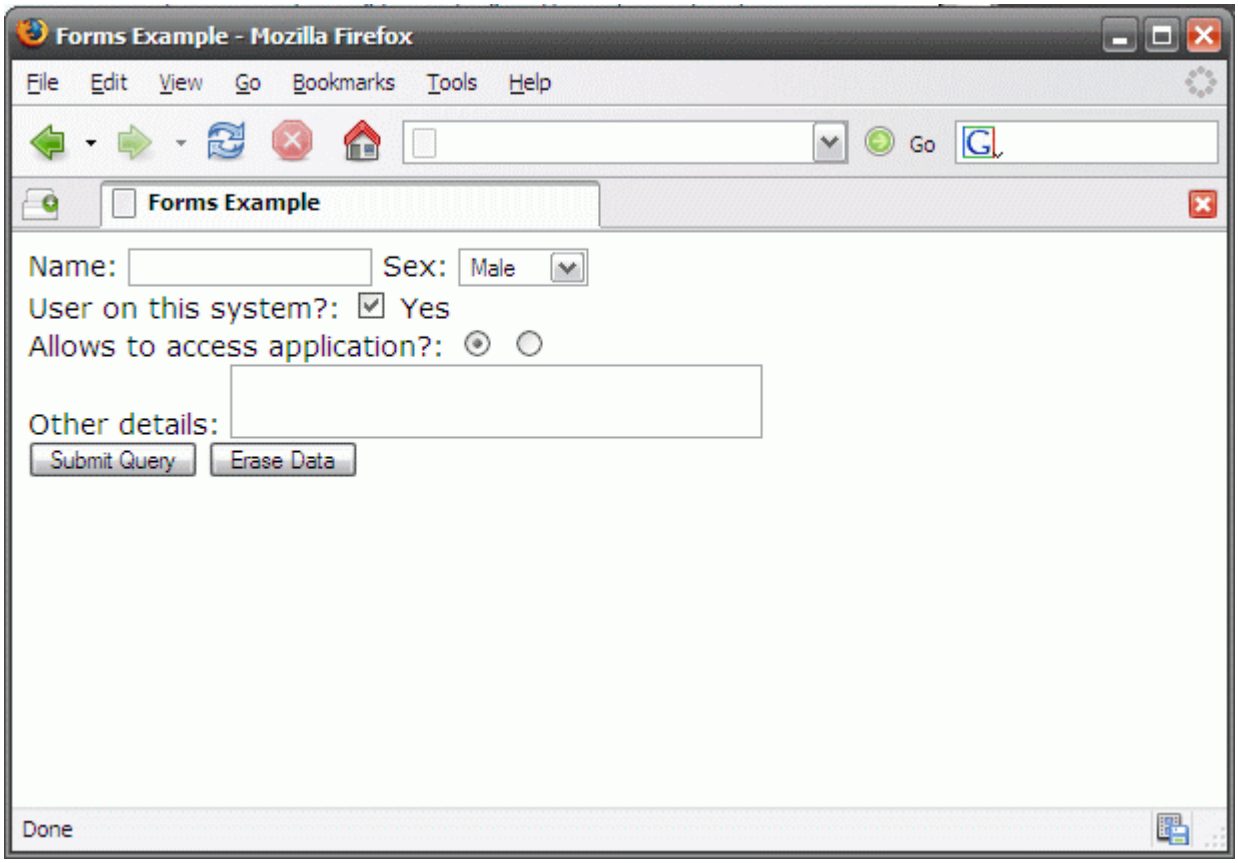

Fig. 3.5 Sample HTML output using Form tags

A host of HTML editors are available for generating documents with the HTML tags. For instance, **Nvu** (pronounced "N-view"), wysiwig HTML editor, is meant to be an open source equivalent to proprietary software like Microsoft FrontPage and Macromedia Dreamweaver. It is also one of the primary wysiwyg editors for Linux and has got an integrated CSS editor. It is designed to make it easy for users who are not computer savvy and does not require knowledge of HTML or CSS.

3.4 Cascading Style Sheets (CSS)

Separation of style and content is a methodology of web design in which the markup language (e.g., HTML) of a web page contains the structure and meaning of the document, but does not define the visual layout, i.e., the style with which the page is to be displayed on the screen. The style information is defined in an external style sheet using a special language like CSS (Cascading Style Sheets). This approach has some advantages when compared with the traditional web design methodology where the content and style are defined by the page's HTML itself. Some of the advantages of separating style from content are:

- Speed -- Once a style sheet has been downloaded into the browser's cache it can be used for all pages that use that style sheet.
- Maintainability – A single style sheet can be used for the entire site. Any changes to that style sheet can propagate to all pages. New pages can also be added easily without having to add layout information to them.
- Accessibility – Using style sheets, a page can be tweaked to display properly in various browsers much more easily than tweaking the markup of a page. Style sheets also allow the page to degrade gracefully, allowing the site to display its contents even in text only browsers or screen readers. In such cases, the browser or reader can ignore the styling information and show just the content.
- Customization -- Using style sheets allows the site author to specify multiple style sheets according to the target audience. Users can also specify their own style sheets to suit their needs.
- Consistency – Using style sheets means the styling of the entire website is consistent. Authors do not have to worry about the styling while adding content inputs..
- Portability – Style sheets allow the document to be used for an entirely different presentation medium (like a printer) with a carefully prepared style sheet.

Style sheets can be written in the following manner:

a) In line style sheet : written directly inside the HTML file

```
<style type=”text/css”>
h1 { color: green; }
</style>
```

b) Linked style sheet : written in a separate .css file and reference / linked into the HTML file

```
<style type=”text/css”>
@import url(/css/stylesheet.css)
</style>
```

c) In-place styles : written directly into the HTML element

```
<table style=”border: 1px solid red; padding: 5pt
```

CSS is a formatting language for documents maintained by the W3C. It is widely used for applying styles to web pages. CSS allows for separation of style and content. CSS allows a document's style to be influenced by multiple style sheets. One style sheet could inherit or "cascade" from another resulting in the name Cascading Style Sheets. New styles can “overlay” existing styles in specific contexts. CSS uses simple English keywords to specify the names of various style properties.

CSS can be applied to a document from various sources.

- Web page authors can provide their style sheets by various methods like:
  external style sheets which are separate CSS files referenced from the web page
  embedded CSS inside the HTML document itself in line styles inside a HTML tag using the "style" attribute
- Users can specify a style sheet to act as an override to be applied to all documents
- User agent style sheet, which is a browser's default style sheet, can be applied to all web pages.

A style sheet consists of a list of **rules**. Each rule consists of one or more comma-separated **selectors** and a **declaration block**. A declaration-block consists of a list of semicolon-separated **declarations** in curly braces. Each declaration itself consists of a **property**, a colon (**:**) then a **value**.

The Style sheet syntax has the following features:

- keywords vs values, e.g. red vs "red"
- definition blocks consist of a tag, class or id, and then a set of styles or properties enclosed in curly brackets.
- White spaces have no specific meaning and comments are enclosed in /*..*/ blocks.
- Colors can be specified using standard names or using HEX or RGB values

  e.g. Color: red (named color)
  color: #3066b0 (HEX value)
  color: rgb(20, 45, 213) (RGB values)
- Margins are the amount of space of an element's bounding box will have with the next element. The top, bottom, left and right margins are set by default, but can be specified individually.
- Padding refers to the internal spacing the element uses within its bounding box.
- Border allows one to define a border for an element. It can be all around or only for specific direction(s). The border type can be one of : solid, dotted or dashed.
- **color** sets the foreground color of an element while the **background-color** allows us to set the background color on the element.

HTML elements can use an addition **id** or **class** attribute to given them a unique identity in the document tree. Classes can be specified as :

<a class="toplink" href="/top/link1.html"> Top L
<h1 class="heading"> Some Heading </h1>

The same class can be used for multiple elements if the same style is applied for all of them.

IDs are used to uniquely identify an element. This is useful if the style for a specific element cannot be defined in a generalized manner.

 This is Section Two
<td id="imp-cell" Important Table Cell </td>

A few examples:

```
p {
   font-family: "Courier";
}
```

indicates that all paragraphs have the default font family to Courier.

```
h2 {
   font-size: 110%;
   color: red;
   background: white;
}
```

indicates that H2 headings are 110% the normal size with a red font on a white background.

```
a {
      color: blue;
}
a:hover {         /* special rule while the mouse hovers over links */
      color: red;
}
```

The above two rules state that hyper links in the document are shown in blue.
Their color changes to red when the mouse is hovering over them.

CSS Selectors

In CSS, pattern matching rules determine which style rules apply to elements in the document tree. These patterns, called selectors, may range from simple element names to rich contextual patterns. If all conditions in the pattern are true for a certain element, the selector matches the element. In the examples given above, p, h2, a, a:hover are the selectors. Same style for multiple tags can be defined by grouping them together and applying a common style to them in one go. It can also be done by matching the name of the "document language element type" wherein the style applies to every such element in the document.

The following table taken from the W3C's recommendations for CSS list the various patterns and their meanings.

Pattern	**Meaning**
*	Matches any element.
E	Matches any E element (i.e., an element of type E).
E F	Matches any F element that is a descendant of an E element.
E > F	Matches any F element that is a child of an element E.
E:first-child	Matches element E when E is the first child of its parent.
E:link E:visited	Matches element E if E is the source anchor of a hyper link of which the target is not yet visited (:link) or already visited (:visited).
E:active E:hover	Matches E during certain user actions.

E:focus	
E:lang(c)	Matches element of type E if it is in (human) language c (the document language specifies how language is determined).
E + F	Matches any F element immediately preceded by a sibling element E.
E[foo]	Matches any E element with the "foo" attribute set (whatever the value).
E[foo="warning"]	Matches any E element whose "foo" attribute value is exactly equal to "warning".
E[foo~="warning"]	Matches any E element whose "foo" attribute value is a list of space-separated values, one of which is exactly equal to "warning".
E[lang|="en"]	Matches any E element whose "lang" attribute has a hyphen-separated list of values beginning (from the left) with "en".
DIV.warning	*Language specific.* (In HTML, the same as DIV[class~="warning"].)
E#myid	Matches any E element with ID equal to "myid".

Table 3.1 CSS patterns and their meanings

Using CSS

To use a CSS style sheet, the CSS code has to be saved in a file such as *example.css* and then linked to or imported from HTML web pages using one of the following two formats:

```
<link rel="stylesheet" href="example.css" type="text/css" >
or

<style type="text/css">
  @import "example.css";
</style>
```

CSS styles may also be specified in the `<head>` tag or attached to a specific element via the `style` attribute. For a spectacular example of CSS in action, the reader can visit http://www.csszengarden.com/.

In recent times, CSS has become increasingly popular for laying out blocks of text on a page. Designers can have “fluid” layouts which are more flexible and amenable to different screen sizes, browsers, display and input devices. Complex and elaborate web pages can be designed entirely using CSS to the formatting while the HTML tags are used just to define logical blocks of the

content.

3.5 XHTML

XHTML is a markup language which has the same expressive power as HTML but with a stricter syntax. XHTML is designed as a successor to HTML. There are two current versions of XHTML:

- XHTML 1.0 is HTML4 reformulated as an XML application, and is backwards-compatible with HTML in limited cases.
- XHTML 1.1 is a modular version of XHTML that is not backwards-compatible with HTML.

XHTML was developed to deliver content to many devices. While web browsers can easily display HTML not all devices can read the complex HTML syntax. XHTML documents are to be well formed. They are not required to be valid. Since XHTML is an XML application, an XHTML document can contain elements from different XML name spaces (such as MathML and SVG).

XHTML 1.0

It is a reformulation of HTML 4.01 as an XML application. There are three different "flavors" of XHTML 1.0, each equal in scope to their respective HTML 4.01 versions : XHTML 1.0 Strict, XHTML 1.0 Transitional., and XHTML 1.0 Frameset.

XHTML 1.0 Strict

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

The Strict declaration is the least forgiving. This is the preferred DOCTYPE for new documents. Strict documents tend to be streamlined and clean. All formatting will appear in Cascading Style Sheets rather than the document itself.

XHTML 1.0 Transitional

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

This declaration is intended as a halfway house for migrating legacy HTML documents to XHTML 1.0 Strict. The W3C encourages authors to use the Strict DOCTYPE for new documents. (The XHTML 1.0 Transitional DTD refers readers to the relevant note in the HTML4.01 Transitional DTD.)

This DOCTYPE does not require CSS for formatting although it is recommended.

XHTML 1.0 Frameset

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

For creating a page with frames, this declaration is appropriate. However, since frames are generally discouraged when designing Web pages, this declaration should be used rarely.

XHTML 1.1

XHTML 1.0 is a suitable markup language for most purposes. It provides the option to separate content and presentation, which fits the needs of most Web authors. XHTML 1.1 enforces the separation of content and presentation. All deprecated elements and attributes have been removed. It also removes two attributes that were retained in XHTML 1.0 purely for backwards-compatibility. The lang attribute is replaced by xml:lang and name is replaced by id. Finally it adds support for Ruby text found in East Asian documents.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

The modularization of XHTML, or XHTML m12n, provides suggestions for customizing XHTML, either by integrating subsets of XHTML into other XML applications or extending the XHTML element set. The framework defines two processes:

- How to group elements and attributes into "modules"
- How to combine modules to create new markup languages

The resulting languages, which the W3C calls "XHTML Host Languages", are based on the familiar XHTML structure but specialized for specific purposes. XHTML 1.1 is an example of a host language. It was created by grouping the different elements available to XHTML. XHTML variations, while possible in theory, have not been widely adopted. There is continuing work being done to develop host languages but their details are beyond the scope of this discussion.

Converting a HTML document into XHTML

Following some simple rules enables us to transform a HTML document into an XHTML document.

- Documents must be well-formed
  - Tags must be properly nested
  - Elements must be closed
- Tags must be lowercase
- Attribute names must be lowercase
- Attribute values must be quoted
- Attributes cannot be minimized
- The name attribute is replaced with the id attribute (in XHTML 1.0 both name and id should be used with the same value to maintain backwards-compatibility).
- Plain ampersands are not allowed

- Scripts and CSS must be escaped(enclose them within the tags <![CDATA[and]]>) or preferably moved into external files.

When is the conversion appropriate

Converting an HTML document into XHTML is easy, but tedious. Before embarking on this course, it is important to analyze whether it is necessary to do the conversion.

- Do you want your pages to be easily viewed over a nontraditional Internet-capable device, such as a PDA or Web-enabled telephone? Will this be a goal of your site in the future? XHTML is the language of choice for Web-enabled portable devices. Now may be a good time for you to commit to creating an all-XHTML site.

- Do you plan to work with XML in the future? If so, XHTML may be a logical place to begin. If you head up a team of designers who are accustomed to using HTML, XHTML is a small step away. It may be less intimidating for beginners to learn XHTML than it is to try teaching them all about XML from scratch.

- Is it important that your site be current with the most recent W3C standards? Staying on top of current standards will make your site more stable and help you stay updated in the future, as you will only have to make small changes to upgrade your site to the newest versions of XHTML as they are approved by the W3C.

- Will you need to convert your documents to another format? As a valid XML document, XHTML can utilize XSL to be converted into text, plain HTML, another XHTML document, or another XML document. HTML cannot be used for this purpose.

If the answer is yes to any of the above questions, then one should probably convert the Web site to XHTML.

MIME Types

XHTML 1.0 documents should be served with a MIME Type of application/xhtml+xml to Web browsers that can accept this type. XHTML 1.0 may be served with the MIME type text/html to clients that can not accept application/xhtml+xml provided that the XHTML complies with the additional constraints of the XHTML 1.0 specification.

It should be checked that the XHTML documents are served correctly to browsers that support application/xhtml+xml, e.g. Mozilla Firefox. 'Page Info' should be used to verify that the type is correct.

XHTML 1.1 documents are often not backwards compatible with HTML and should not be served with a MIME type of text/html.

XHTML Sample

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>XHTML 1.0 Example</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  </head>
  <body>
    <p>This is a tiny example of an <abbr title="Extensible HyperText
Markup Language">XHTML</abbr> 1.0 Strict document.</p>
  </body>
</html>
```

XHTML Mobile Profile

- XHTML Mobile Profile (XHTML MP) is a XHTML vocabulary designed specifically for mobile phones and other resource-constrained devices.
- XHTML-MP is derived from XHTML 1.1 Basic Profile by adding XHTML Modules, with later versions of the standard adding more modules.
- The XHTML MP 1.1 DTD is currently in the process of being finalized.
- Devices supporting older versions of WAP can be served by transforming the XHTML MP document into WAP.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.0//EN"
"http://www.wapforum.org/DTD/xhtml-mobile10.dtd">
```

- Styling done using Wireless CSS (WCSS).
- MIME Type: application/vnd.wap.xhtml+xml

3.6 XML

XML is a W3C-recommended general-purpose markup language. XML is useful for creating special-purpose markup languages, capable of describing many different kinds of data. XML is a simplified subset of Standard Generalized Markup Language (SGML). It provides a text-based means to describe and apply a tree-based structure to information. An XML document contains one or more entities each of which have tags. Users define the meanings for tags and attributes. This allows users to create documents that contain more explicit information about the content.

XML offers a number of advantages. While HTML offers display, XML offers structure and content. It is human and machine readable format which is flexible and simple. Industry specific documents can be standardized and this happens often in practice. Using the features of XML websites can provide more information for search engines resulting in more accurate results. A single XML document can be rendered in many different ways using Extensible Style sheet Language(XSL).

XML History

XML was developed to meets the needs for a general purpose markup language for the web which was easier to use than SGML. The new standard had to be more expressive than HTML while not becoming too complex. The initial XML Working Group's goals were for the new standard to have Internet usability, general-purpose usability, SGML compatibility, facilitation of easy development of processing software, minimization of optional features, legibility, formality, conciseness, and ease of authoring.

- Working on the standard began in 1995.
- XML 1.0 became a W3C Recommendation on February 10, 1998.
- XML 1.1, a variant of XML that encourages more consistency in how characters are represented and relaxes restrictions on names, allowable characters, and end-of-line representations was released on February 4, 2004.
- XML 1.0 Fourth Edition and XML 1.1 Second Edition are considered current versions of XML.

XML Syntax

XML documents start with an XML declaration which contain the version information – usually 1.0 and any encodings and external dependencies.

```
<?xml version="1.0" encoding="UTF-8"?>
```

The basic syntax for an element in XML is of the form

```
<name attribute="value">content</name>
```

After the XML declaration, there are a number of nested elements, some of which have attributes and content. An element usually consist of two tags, a start and an end tag enclosing text and other elements. Start tag consists of a name surrounded by angle brackets, e.g. `<book>` The end tag consists of the same name surrounded by angle brackets, but with a forward slash preceding the name. `</book>`

Everything that appears between the start and end tag of an element is its content. Some elements can have the optional attributes – name-value pairs included in the start tag after the element name. Attribute values must be quoted using single or double quotes. Each attribute name can occur only once in an element.

```
<book condition=”new”>Glimpses of World History</book>
```

Each XML document can have exactly one top-level root element. There is a special syntax for elements with empty content. In an empty element tag a slash follows immediately after the element name, e.g. <marker/>

Special characters are represented in XML using entity references and numeric character references.

An entity reference is a placeholder that represents the entity. Consists of the entity name preceded by & and ending with a ;

XML has five predeclared entities :

- & (&)
- < (<)
- > (>)
- ' (')
- " (")

When viewed in a suitable viewer the entities are displayed.

Numeric character references look like entities, but instead of a name, they contain the "#" character followed by a number. The number (in decimal or "x"-prefixed hexadecimal) represents a Unicode code point, and is typically used to represent characters that are not easily encodable.

Correctness of an XML document

XML is much more strict about the interpretation of tags than HTML. While web browsers will accept malformed HTML and try to display them to the best of their ability, such behavior is not allowed for XML. Any program which deals with XML uses an XML parser which checks the XML before using it. There are strict rules for the XML parsers to follow before declaring the XML as suitable for further processing.

For an XML document to be correct, it must be:

- Well Formed – the document conforms to all of XML' s syntax rules. If a document is not well formed it is not an XML document and XML parsers will refuse to parse the document.
- Valid – the document in addition to be well formed conforms to a particular set of user defined rules, or XML schemas that describe the structure of the document.

Not all documents are required to be valid. Some applications might require that the XML document be well formed. They might not care if the document is valid or not. As mentioned, the validity of an XML document is tested by checking whether it conforms to certain defined rules. These rules are known as XML schemas. The users have to supply the XML schemas for the particular application. There are a few different XML schema languages available. Let us briefly look at a few of them.

Document Type Definition (DTD)

DTD is the oldest XML schema format. A Document Type Definition (DTD) defines the syntax,

structure and vocabulary that is allowable in the corresponding XML document. The DTD is a simple text file which contains instructions for how elements relate to one another within the document and which attributes may be used with which elements.

DTDs have a number of limitations. They don't support some of the newer features of XML like name spaces. They cannot capture all aspects of some XML documents and they use a custom non-XML syntax to describe the schema

XML Schema languages

XML Schema Definition (XSD) is the W3C sponsored successor to DTD s.

- Written in XML and much more powerful.
- Very verbose and complex.

RELAX NG

- Simpler than XSD.
- Both an XML and a compact non-XML syntax exist.
- Well defined way to translate between the two forms.

Processing XML files

XML parsers which process XML provide two API s to work with XML files.

- Simple API for XML (SAX) – an event driven api where documents are read serially.
- Document Object Model (DOM) – an api which construct a node tree to process XML.

The selection of the API to use depends on the application. SAX is simpler and is better if complex XML processing is not required. DOM is more heavy-weight but is more suitable for complex document processing.

Another method of processing XML is through XML Databinding. XML Databinding is having XML available as a custom programming language data structure. A filter in the Extensible Stylesheet Language (XSL) family can transform an XML file for displaying or printing. As discussed in a later section, an XML document has no information on how to present the data. It needs either a CSS or an XSL style sheet for presentation.

3.7 Xpath

XPath is the special syntax used in XML to refer to elements in an XML document. Although it is used for referring to nodes in an XML tree, Xpath itself is not written in XML since it is a very

cumbersome task to specify path information in XML. Any characters that form XML syntax would need to be escaped so that it is not confused with XML when being processed. XPath is also very succinct, allowing you to call upon nodes in the XML tree with a great degree of specificity without being unnecessarily verbose.

XPath works on the XML tree which is similar to the HTML content tree. Each element or node of the tree has some attributes and relations with other nodes in the tree. For example a node can be parent to other nodes while being the child of another node. It may also have some sibling nodes. Using this, XPath builds the concept of Location Paths. A location path is the series of location steps taken to reach the node/nodes being selected. Location steps are the parts of XPath statements separated by / characters. They are one step on the way to finding the nodes you would like to select.

Location steps are comprised of three parts:

1. an axis (child, parents, descendant, etc.)
2. a node test (name of a node, or a function that retrieves one or more nodes)
3. a series of predicates (tests on the retrieved nodes that narrow the results, eliminating nodes that do not pass the predicates test).

Thus, in a location path, each of its location steps returns a node-list. If there are further steps on the path after a location step, the next step is executed on all the nodes returned by that step. Location Paths can be in Abbreviated XPath and Unabbreviated XPath.

To select the parent of the current node which is being processed we have:

- Abbreviated XPath: ..
- Unabbreviated XPath: parent::node()

Paths can be absolute (starting from the root element) or relative (starting from the current node).

Using the two concepts results in four different types of Location Paths:

- Abbreviated Relative Location Paths- Use of abbreviated syntax while specifying a relative path.
- Abbreviated Absolute Location Paths- Use of abbreviated syntax while specifying an absolute path.
- Unabbreviated Relative Location Paths- Use of unabbreviated syntax while specifying a relative path.
- Unabbreviated Absolute Location Paths- Use of unabbreviated syntax while specifying an absolute path.

XPath also provides some predicates and functions. These allow for the predicate to be evaluated for such things as true/false and count functions. For example:

```
/book/chapter[position()=3]
```

returns the third node denoting the third chapter of the book. For something to be returned, the current <book> element must have at least 3 <chapter> elements.

This short discussion of Xpath is to enable the reader to understand Xpath statements which occur in XML Stylesheets.

3.8 XSL

An XML document has no information on how to present the data. So it needs either a CSS or XSL style sheet for presentation.

```
<?xml-stylesheet type="text/css" href="myStyleSheet.css"?>
<?xml-stylesheet type="text/xsl" href="myTransform.xslt"?>
```

XSL Transformation (XSLT)

XSLT allows for manipulating and sorting the content of an XML document including wholesale reordering. It can be used to change a document from one XML vocabulary to another. XSLT is commonly used to transform XML documents into XHTML documents which can then be displayed on browsers. XSLT is built on top of XPath.

An XSLT style sheet is an XML document, complete with elements and attributes. It has two kinds of elements, top-level and instruction. Top-level elements fall directly under the style sheet root element. Instruction elements represent a set of formatting instructions that dictate how the contents of an XML document will be transformed. During the transformation process, XSLT analyzes the XML document, or the source tree, and converts it into a node tree, a hierarchical representation of the entire XML document, also known as the result tree. Each node represents a piece of the XML document, such as an element, attribute or some text content. The XSL style sheet contains predefined “templates” that contain instructions on what to do with the nodes. XSLT will use the match attribute to relate XML element nodes to the templates, and transform them into the result document.

XSLT can be used to transform an XML source into many different types of documents. XHTML is also XML, if it is well formed, so it could also be used as the source or the result. However, transforming plain HTML into XML won't work unless it is first turned into XHTML so that it conforms to the XML 1.0 recommendation. Here is a list of all the possible type-to-type transformations performed by XSLT:

	XML	**XHTML**	**HTML**	**Text**
XML	X	X	X	X
XHTML		X	X	X

The output element in the style sheet determines how to output the result tree. This element is optional, but it allows you to have more control over the output. If you do not include it, the output method will default to XML, or HTML if the first element in the result tree is the <html> element.

XSL-FO

XSL-FO stands for Extensible Stylesheet Language Formatting Objects and is a language for formatting XML data. When it was created, XSL was originally split into two parts, XSL and XSL-FO. Both parts are now formally named XSL. XSL-FO documents define a number of rectangular areas for displaying output. XSL-FO is used for the formatting of XML data for output to screen, paper or other media, such as PDF format.

XSLT Example

Source Document

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="example.xsl"?>
<Article>
  <Title>My Article</Title>
     <Author>Senthil</Author>
      <Body>This is my article text.</Body>
</Article>
```

XSLT Stylesheet

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="text"/>
<xsl:template match="/">
    Article - <xsl:value-of select="/Article/Title"/>
    Author: <xsl:apply-templates select="/Article/Author"/>
</xsl:template>
<xsl:template match="Author">
    - <xsl:value-of select="." />
</xsl:template>
</xsl:stylesheet>
```

Browser Output

```
Article - My Article
Author:
- Senthil
```

3.9 Dynamic Web Pages

A dynamic web page is one which utilizes client-side scripting (such as JavaScript) to effect changes in variables of the presentation definition language (such as CSS). This, in turn, will effect the look and function of the otherwise "static" HTML page content, *after* the page has been fully loaded and during the viewing process. No information is sent or received from the server after the page has been downloaded. Dynamic HTML (DHTML) uses JavaScript as the driving force for its functioning. Javascript interacts with the web page using the W3C's Document Object Model (DOM). These two topics, Javascript and DOM, will be discussed in the following sections (3.10 and 3.11). In the final section 3.12, we will also discuss AJAX (Asynchronous JavaScript and XML), which deals with two powerful browser features to make requests to the server without reloading the page, and to parse and work with XML documents.

3.10 Javascript

History

Javascript was developed by the Netscape Corporation in 1995/96 as a means for scripting the Netscape 2 browser. It was originally called Livescript but the name was changed to Javascript due to the rising popularity of Java. In spite of its name there is no relation between Java and Javascript. Microsoft provided a mostly compliant version called Jscript with IE 3. Netscape sent the language for standardization to ECMA International in 1997. The language was standardised as ECMAscript in 1997. The version 3 of the standard was completed in 1999 and the work for a new version 4 was taken up thereafter. The relative stability of the standard for a long time means that there are many mature implementations available and plenty of tools to work with it.

Where is it used ?

Javascript was designed to run as a scripting language within a host environment and so lacks the concept of input and output. Web browsers are the most common hosts for Javascript. Many applications also use javascript for their scripting needs. The examples in this book were tested using the SpiderMonkey Javascript engine from www.mozilla.org. Packages can be found in various linux distributions.

Javascript types

Javascript has the following types: *Number, String, Boolean, Object (Function, Array, Date, RegExp), Null* and *Undefined.* The various types are briefly explained below.

Numbers

Javascript specification defines numbers as “double-precision 64-bit format IEEE 754 values”. All standard numeric operations like addition, subtraction, multiplication, division etc. are supported. More advanced mathematical functions are handled by the inbuilt Math object.

```
D = Math.PI * r * r
```

String to integer conversion is done using the parseInt function. The function takes the base of the number as an optional second argument.

```
> parseInt(“123”, 10)
  123
> parseInt(“11”,2)
  3
> parseInt(“hello”, 10)
  NaN
```

Strings

Strings in Javascript are sequences of Unicode characters.

```
> "hello".length
  5
> "hello".charAt(0)
  h
> "hello, world".replace("hello", "goodbye")
  goodbye, world
> "hello".toUpperCase()
  HELLO
```

Other Types

- Null is an object of type object that indicates a deliberate non-value.
- Undefined is an object of type undefined that indicates an uninitialized value.
- Boolean type, with possible values of true and false.
- Any value can be converted to a boolean according to the following rules:
  false, 0, the empty string(“”), Nan, null and undefined all become false.
  All other values become true

Variables

New variables in Javascript are declared using the var keyword:

```
var a;
var name = "simon";
```

A variable declared without any value is taken to be of type undefined.

Operators

JavaScript's numeric operators are +, -, *, / and % - which is the remainder operator. Values are assigned using =, and there are also compound assignment statements such as += and -= - these extend out to x = x operator y.

```
x += 5
```

```
x = x + 5
```

The operators ++ and -- are used to increment and decrement respectively. These can be used as prefix or postfix operators.

The + operator also does string concatenation:

```
> "hello" + " world"
hello world
```

While adding a string to a number (or other value) everything is converted in to a string first.

```
> "3" + 4 + 5
345
> 3 + 4 + "5"
75
```

Adding an empty string to something is a useful way of converting it.

Comparisons in JavaScript can be made using <, >, <= and >=. These work for both strings and numbers. Equality is a little less straightforward. The double-equals operator performs type coercion if given different types, with sometimes interesting results:

```
> "dog" == "dog"
true
> 1 == true
true
```

To avoid type coercion, the triple-equals operator should be used:

```
> 1 === true
false
> true === true
true
```

There are also != and !== operators and JavaScript allows has bitwise operations as well.

Control structures

JavaScript has a similar set of control structures to other languages in the C family. Conditional statements are supported by if and else and they can be chained together if needed:

```
var name = "kittens";
if (name == "puppies") {
  name += "!";
} else if (name == "kittens") {
  name += "!!";
} else {
```

```
  name = "!" + name;
}
name == "kittens!!"
```

JavaScript has while loops and do-while loops. The first is good for basic looping; the second for loops where you wish to ensure that the body of the loop is executed at least once:

```
while (true) {
  // an infinite loop!
}

do {
  var input = get_input();
} while (inputIsNotValid(input))
```

JavaScript's for loop is the same as that in C and Java: it lets you provide the control information for your loop on a single line.

```
for (var i = 0; i < 5; i++) {
  // Will execute 5 times
}
```

The && and || operators use short-circuit logic, which means they will execute their second operand dependent on the first. This is useful for checking for null objects before accessing their attributes:

```
var name = o && o.getName();
```

Or for setting default values:

```
var name = otherName || "default";
```

JavaScript has a tertiary operator for one-line conditional statements:

```
var allowed = (age > 18) ? "yes" : "no";
```

The switch statement can be used for multiple branches based on a number or string:

```
switch(action) {
    case 'draw':
        drawit();
        break;
    case 'eat':
        eatit();
        break;
    default:
        donothing();
}
```

If you don't add a break statement, execution will "fall through" to the next level. This is very rarely what you want - in fact, it's worth specifically labelling deliberate fallthrough with a comment if you really meant it to aid debugging:

```
switch(a) {
    case 1: // fallthrough
    case 2:
        eatit();
        break;
    default:
        donothing();
}
```

The default clause is optional. You can have expressions in both the switch part and the cases if you like; comparisons take place between the two using the === operator:

```
switch(1 + 3):
    case 2 + 2:
        yay();
        break;
    default:
        neverhappens();
}
```

Objects

JavaScript objects are simply collections of name-value pairs. The "name" part is a JavaScript string, while the value can be any JavaScript value - including more objects.

```
var obj = {
name: “MyObject”;
details: {
size: tiny;
      }
};

> obj.details.size
    tiny
```

Arrays

Arrays in JavaScript are a special type of object. They are similar to other objects but have a special property, 'length' of an array which is always one more than the highest index in the array.

```
> var a = ["dog", "cat", "hen"];
> a.length
  3
```

To append an item to an array

```
a[a.length] = item;
```

Arrays come with a number of methods. Some of them are:

```
a.concat(item, ..)
      -- concat returns a new array with the items added on to it.
a.pop()
      -- pop removes and returns the last item
a.push(item, ..)
      -- push adds one or more items to the end (like our ar[ar.length] idiom)
a.slice(start, end)
      -- slice returns a sub-array
a.sort(cmpfn)
      -- sort takes an optional comparison function
a.splice(start, delcount, [item]..)
      -- splice lets one modify an array by deleting a section and replacing it
with more items
a.unshift([item]..)
      -- unshift prepends items to the start of the array
```

Functions

Functions are a core component of Javascript. A JavaScript function can take 0 or more named parameters. The function body can contain as many statements as you like, and can declare its own variables which are local to that function. The return statement can be used to return a value at any time, terminating the function. If no return statement is used (or an empty return with no value), JavaScript returns undefined. Functions can also be called recursively.

```
function avg() {
    var sum = 0;
    for (var i = 0, j = arguments.length; i < j; i++) {
        sum += arguments[i];
    }
    return sum / arguments.length;
}
> avg(2, 3, 4, 5)
3.5
```

Anonymous Functions

Javascript allows a full function definition anywhere where there can be an expression.

```
> var a = 1;
> var b = 2;
> (function() {
   var b = 3;
   a += b;
  })();
> a
  4
> b
  2
```

Functions have access to an additional variable inside their body called arguments. It is an array-like object holding all of the values passed to the function. arguments.callee always refers to the current function. This feature can be used to call anonymous functions recursively.

```
function counter() {
    if (!arguments.callee.count) {
        arguments.callee.count = 0;
    }
    return arguments.callee.count++;
}

> counter()
0
> counter()
1
> counter()
2
```

Custom Objects

Javascript allows us to create custom objects.

```
function Person(first, last) {
this.first = first;
this.last = last;
}

Person.prototype.fullName = function() {
return this.first + ' ' + this.last;
}

Person.prototype.fullNameReversed = function() {
return this.last + ', ' + this.first;
}
```

The 'this' keyword when used inside a function refers to the current object.

Person.prototype is an object shared by all instances of Person. It forms part of a lookup chain (that has a special name, "prototype chain"). During any attempt to access a property of Person that isn't set, JavaScript will check Person.prototype to see if that property exists there instead. Anything assigned to Person.prototype becomes available to all instances of that constructor via the this object.

JavaScript allows modification of an object's prototype at any time. Can add extra methods to

existing objects at runtime.

```
> s = new Person("Senthil", "Anand");
> Person.prototype.firstNameCaps = function() {
  return this.first.toUpperCase()
  }
> s.firstNameCaps()
  SENTHIL
```

The keyword 'new' is strongly related to 'this'. It creates a brand new empty object, and then calls the function specified, with 'this' set to that new object. Functions that are designed to be called by 'new' are called constructor functions. Common practice is to capitalize these functions as a reminder to call them with new.

Inner functions

JavaScript function declarations are allowed inside other functions. An important detail of nested functions in JavaScript is that they can access variables in their parent function's scope:

```
function aRandomFunction() {
    var a = 1;
    function oneMoreThanA() {
        return a + 1;
    }
    return oneMoreThanA();
}
```

This provides a great deal of utility in writing more maintainable code. If a function relies on one or two other functions that are not useful to any other part of your code, you can nest those utility functions inside the function that will be called from elsewhere. This keeps the number of functions that are in the global scope down, which is always a good thing.

This is also a great counter to the lure of global variables. When writing complex code it is often tempting to use global variables to share values between multiple functions - which leads to code that is hard to maintain. Nested functions can share variables in their parent, so you can use that mechanism to couple functions together when it makes sense without polluting your global namespace - 'local globals' if you like. This technique should be used with caution, but it's a useful facility to have.

3.11 Document Object Model (DOM)

DOM is an API for HTML and XML documents which is maintained by the W3C. DOM connects web pages to programming languages by providing a structured representation of HTML and XML documents. Using DOM the developer can modify documents on the fly.

As the name DOM implies the document is considered as a collection of objects. Web browsers

allow access to these objects and their attributes via scripting languages. Javascript is the most widely used language for working with DOM. Bindings for many languages exist – W3C has links for bindings in C++, PHP, Python and other languages in addition to Javascript.

History of W3C DOM

Dynamic HTML which had pop-up menus, layer effects etc was developed to get some interactivity to the WWW. But DHTML was not cross browser compatible; that is, DHTML coded for Netscape Navigator would not work with Microsoft Internet Explorer. DHTML needed a standard way of accessing HTML documents to be used widely. W3C came up with W3C DOM based on the DOM of Navigator and Internet Explorer. The various DOM Levels are :

Level 0

The DOM used before the W3C standardization. Varies from browser to browser.

Level 1

The DOM Level 1 specification is separated into two parts: Core and HTML. The Core DOM1 section provides a low-level set of fundamental interfaces that can represent any structured document, as well as defining extended interfaces for representing an XML document. The HTML Level 1 section provides additional, higher-level interfaces that are used with the fundamental interfaces defined in the Core Level 1 section to provide a more convenient view of an HTML document. Interfaces introduced in DOM1 include, among others, the Document, Node, Attribute, Element, and Text interfaces. All interfaces contain attributes and/or methods that can be used to interact with XML and HTML documents.

Level 2

The DOM Level 2 specification contains six different specifications: The DOM2 Core, Views, Events, Style, Traversal and Range, and the DOM2 HTML.

- The DOM2 Core extends the functionality of the DOM1 Core. It also contains specialized interfaces dedicated to XML. Examples of methods introduced in the DOM2 Core include getElementById, and many namespace-related methods.

- The DOM2 Views allows programs and scripts to dynamically access and update the content of a representation of a document. The introduced interfaces are AbstractView and DocumentView.

- The DOM2 Events gives a generic event system to programs and scripts. It introduces the concepts of event flow, capture, bubbling, and cancelation. Some methods here include addEventListener and handleEvent. Several interfaces for dealing with events: EventTarget, EventListener, Event, DocumentEvent, MouseEvent, MutationEvent, etc. However, it does not include an interface for the keyboard events, which will be dealt with in later versions of the DOM.

- The DOM2 CSS, or DOM2 Style, allows programs and scripts to dynamically access and

update the content of style sheets. It has interfaces for Style Sheets, Cascading Style Sheets, CSSRule, CSSStyleDeclaration, the famous getComputedStyle , the many many CSS2Properties, and all the media rules.

- The DOM2 Traversal and Range allow programs and scripts to dynamically traverse and identify a range of content in a document. The DOM2 Traversal provides interfaces like NodeIterator and TreeWalker to easily traverse the content of a document. The DOM2 Range allows the creation, insertion, modification, and deletion of a range of content in a Document, DocumentFragment, or Attr. It can be characterized as selecting all of the content between a pair of boundary-points.

- The DOM2 HTML allows programs and scripts to dynamically access and update the content and structure of HTML documents. It extends the interfaces defined in the DOM1 HTML, using the DOM2 Core possibilities. It introduces the contentDocument property, a useful way to access the document contained in a frame.

Level 3

The DOM Level 3 specification contains five different specifications: The DOM3 Core, Load and Save, Validation, Events, and XPath.

- The DOM3 Core will extend the functionality of the DOM1 and DOM2 Core specs. New methods and properties include adoptNode(), strictErrorChecking, and textContent, to name only a few.

- The DOM3 Load and Save allows programs and scripts to dynamically load the content of an XML document into a DOM document, and serialize a DOM document into an XML document.

- The DOM3 Validation allows programs and scripts to dynamically update the content and the structure of documents while ensuring that the document remains valid, or to ensure that the document becomes valid.

- The DOM3 Events is the extension of the DOM2 Events specification. This specification mainly focuses on keyboard events and how to handle them.

- The DOM3 XPath provides simple functionalities to access a DOM tree using XPath 1.0.

We will look briefly into a few DOM related functions.

DOM example

```
<html>
<title>Hello World</title>
<body onload="window.alert('Hello World from DOM');">
<p>A simple DOM example
</body>
</html>
```

Content Tree of a Document

The structure will be as given in Fig. 3.6.

```
<html>
<head>
<title> Content Tree </title>
</head>
<body>
<h1> What is a Content Tree ? </h1>
</body>
</html>
```

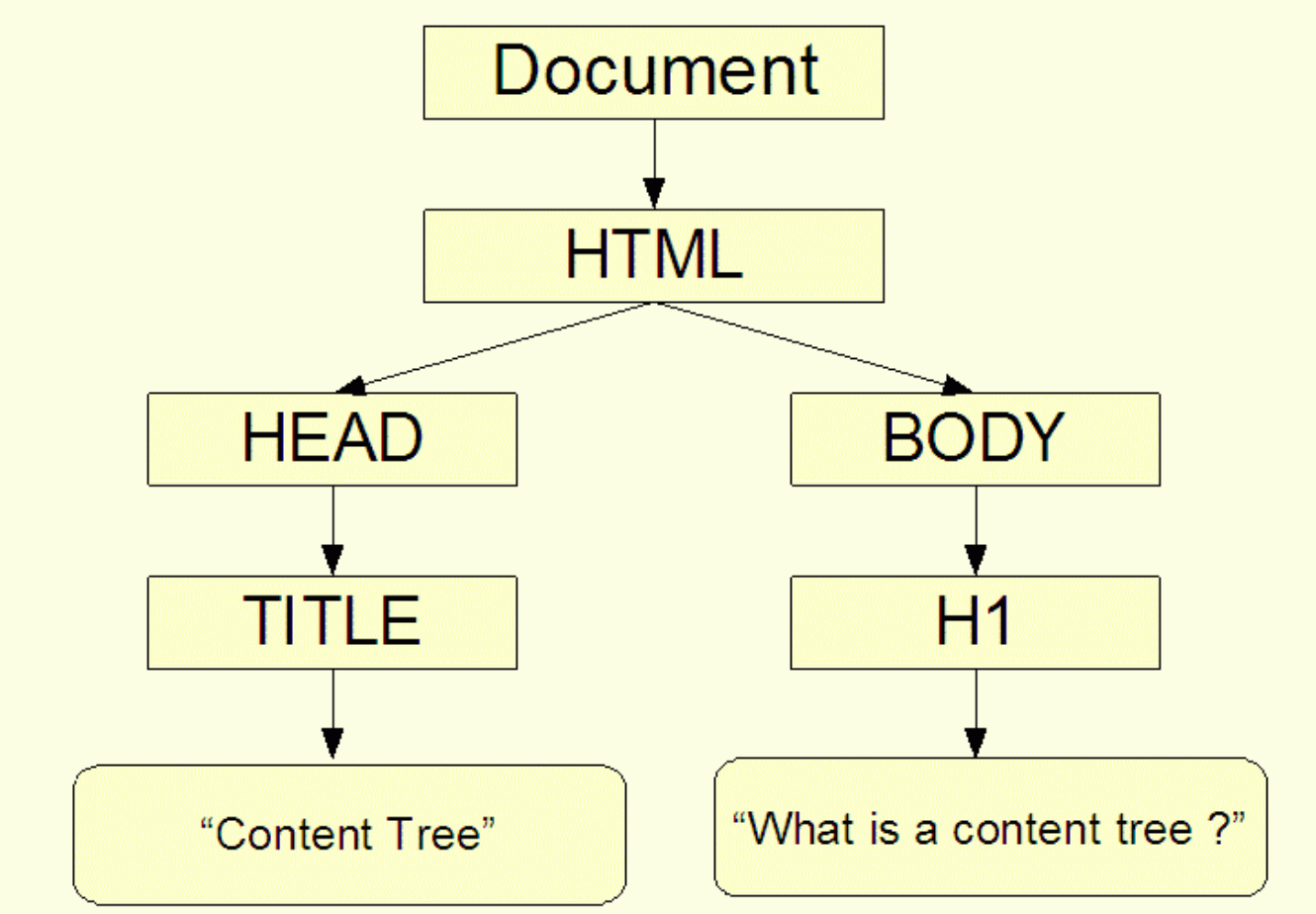

Fig. 3.6 Content tree of a document

Nodes

In DOM all items that compose a document are nodes. Nodes have some common attributes and

methods which can be used to modify the document. Nodes can be either elements or text nodes. In HTML elements are associated with angle-bracketed tags. Elements can also have various attributes and children. Text nodes are different from other elements. Text nodes represent a chunk of text and don't have attributes or children.

Some ways to differentiate between elements and text nodes are:

- nodeType for elements is 1 ,text nodes is 3.
- nodeName for elements is the name of the tag and it is the string "#text" for text nodes.
- nodeValue for elements is null and it is the value of the actual string of the text for text nodes.

Creating a document tree

All documents are represented by the document object. New elements and text nodes are created using methods of the document object.

- createElement(String type) creates an element of the specified type and returns a reference.
- createTextNode(String text) creates a text node with the content specified in text and returns a reference.

```
var myPara = document.createElement("P");
var myText = document.createTextNode("A Text Node");
```

Now myPara and myText contain references to the newly created element and text node. They will not be visible in the browser because they have not been inserted into the document. They have to be assigned to a parent or a document to be visible. Newly created nodes can be attached to documents or parent nodes using the appendChild method. The text node can be inserted to the newly created element by the following code:

```
myPara.appendChild(myText);
```

Now the paragraph has to be added to the body of the document. We need the reference to the body of the document. A reference to an existing node in a document can be obtained using either of the following methods:

- getElementsByTagName which returns a collection of elements it finds with the specified tag.

- getElementById which returns the node with the id attribute specified.

```
var myBody = document.getElementsByTagName("body").item(0);
myBody.appendChild("myPara");
```

Or we can use the following statement if we had assigned an id to the body tag.

```
<body id=”docBody”>
var myBody = document.getElementById(“docBody”);
myBody.appendChild(“myPara”);
```

Getting it all together

```
<script type=”text/javascript”>
function insertNewText() {
var myPara = document.createElement(“P”);
var myText = document.createTextNode(“A Text Node”);
myPara.appendChild(myText);
var myBody = document.getElementsByTagName(“body”).item(0);
myBody.appendChild(“myPara”);
}
</script>
```

Manipulating nodes

There are methods available which can be used to remove a node or replace a node with another.

- removeChild method will remove any child of a node.
- replaceChild method can be used to a child node with a new node.
- insertBefore can be used to insert a child before a specified child.

Element tag attributes can be manipulated using the following methods:

- getAttribute(string name): returns the value of the attribute specified by the name argument.
- setAttribute(string name, string value): adds a new attribute of the specified name and value, or sets the existing attribute to the specified value.

```
var table = document.getElementById("tableMain");
tWidth = table.getAttribute("width");
newTHeight = table.setAttribute(“height”,”300”);
```

hasAttribute() checks whether the given attribute exist for the element.

Element Styles

DOM Level 2 Style can be used to manipulate the CSS properties of an element. Each CSS property of an element is represented by it style object.

```
var doc = document.getElementsByTagName(“body”).item(0);
var bColor = doc.style.backgroundColor;
alert (“Background Color is “ + bColor);
doc.style.backgroundColor = "#0000ff";
```

3.12 AJAX

AJAX is the acronym for "**Asynchronous JavaScript** and **XML",** which is a technique that enables creation of inter active web applications. Most of the web pages are loosely coupled as a combination of pictures, tables and other forms of data. When the user input is made, either as a request to the server or as an input data for further action, it is desirable to avoid reloading of the entire page. This will minimise the reload time and also avoid unnecessary data traffic, thereby increase the web page's interactivity, speed and usability. Pages using Ajax resemble a standalone application and the page can be updated dynamically, enabling a faster response to the user's interaction.

The Ajax technique uses a combination of XTML, CSS, DOM, etc. for

- marking up and styling information (using XHTML and CSS)
- dynamically displaying and interacting with the information presented (DOM accessed with a client-side scripting language)
- exchanging data asynchronously with the web server (e.g., using XMLHttpRequest object)
- transfering data created dynamically by some form of server-side scripting

Ajax has a few limitations too. Since the dynamically created page does not register itself with the browser engine, the "Back" function of the user's browser might not function in the desired manner. It might also be difficult for "bookmarking" the dynamic web page updates. Some level of solutions are being offered for these and other issues to enhance the effectiveness of Ajax technique.

Making a HTTP Request

A HTTP request to the server using JavaScript requires an instance of a class that provides this functionality. The XMLHttpRequest class is created as follows:

```
http_request = new XMLHttpRequest();
http_request.onreadystatechange = nameOfTheFunction;
http_request.open('GET', 'http://www.example.org/some.file', true);
http_request.send(null);
```

Handling the server response

The function which will handle the response should check the value of readyState.

The full list of the readyState values is as follows:

- 0 (uninitialized)
- 1 (loading)
- 2 (loaded)

- 3 (interactive)
- 4 (complete)

The action steps are as follows:

Further processing can be done once the value of the readyState is 4.
The status code of the HTTP server response should also be checked.
A response of 200 OK is the sign to proceed forward.
The data the server has sent can then be processed.
The format of the response can be text or XML.

A complete example

```
<script type="text/javascript" language="javascript">
function makeRequest(url) {
var http_request = false;
if (window.XMLHttpRequest) { // Mozilla, Safari,...
  http_request = new XMLHttpRequest();
    if (http_request.overrideMimeType) {
      http_request.overrideMimeType('text/xml');
      // See note below about this line
    }
} else if (window.ActiveXObject) { // IE
   try {
     http_request = new ActiveXObject("Msxml2.XMLHTTP");
   } catch (e) {
       try {
         http_request = new ActiveXObject("Microsoft.XMLHTTP");
       } catch (e) {}
   }
}

if (!http_request) {
  alert('Giving up :( Cannot create an XMLHTTP instance');
  return false;
}

http_request.onreadystatechange = function()
{ alertContents(http_request); };
http_request.open('GET', url, true);
http_request.send(null);
}

function alertContents(http_request) {
  if (http_request.readyState == 4) {
    if (http_request.status == 200) {
      alert(http_request.responseText);
    } else {
       alert('There was a problem with the request.');
    }
  }
}

</script>
<span
```

```
style="cursor: pointer; text-decoration: underline"
onclick="makeRequest('test.html')">
Make a request
</span>
```

Using an XML response

The test.xml file

```
<?xml version="1.0" ?>
<root>
   I'm a test.
</root>
```

Using XML Response

The request line in the script is changed:

```
onclick="makeRequest('test.xml')">
```

Instead of directly printing out the alert we need to process it.

```
var xmldoc = http_request.responseXML;
var root_node = xmldoc.getElementsByTagName('root').item(0);
alert(root_node.firstChild.data);
```

The above code segment is just a simple example describing the Ajax features wherein a few preliminary checks, e.g., on readyState and HTTP Server response , were made. A robust AJAX application will involve more extensive error checking and ways to degrade gracefully. Usually AJAX websites are not built by hand, but by using frameworks. The frameworks provide many other features on top of XMLHttpRequest. We will look into the frameworks in Chapter 5. However, it should be noted that Ajax is suitable only for certain type of web pages.

In this chapter, the client-side application, View under the Model-View-Controller paradigm, and the user interfaces are discussed. Focussing on the web browser as the interface, creation of static and dynamic web pages is described handling the contents using HTML, etc., and formatting using the example of Cascading Style Sheets. JavaScript, DOM and AJAX are also discussed with reference to dynamic web page handling.

Chapter 4

Data Store Fundamentals and the Dublin Core

In the Model View Controller (MVC) framework, Data Store is the middle layer and refers to the part of the system where all the data are stored. In the present chapter, this topic is studied under the following heads: Data Base Management and PostgresSQL, Server Management and PgAdmin, and Metadata Standards and the Dublin Core.

4.1 Brief Overview of Data Base Management System

A database management system (DBMS) is a software designed to manage databases controlling the organization, storage and retrieval of data therein. MySql and PostgreSQL are widely used in the FOSS arena while Oracle, DB2, Microsoft Access and MS Sql Server are examples of proprietary DBMS systems. There are many excellent books on Data Base Management System [Sudarshan & Kote, 2003; Date, 2000] which is, in fact, studied as a full semester course by itself. The intention here is not to attempt drilling the concepts of DBMS, but to give an outline so as to enable the reader to use PostgresSQL comfortably and follow through the discussions related to the data store layer concepts.

A typical database management system includes :

1. A modelling language to define the schema of each database hosted in the DBMS. The hierarchical, network and relation models are commonly used data models and a DBMS might use one, two or all three of them, besides other methods. The model will depend on the application, the transaction rate and the number of inquiries. Most DBMSs support the *Open Database Connectivity API* which provides a standard way for programmers to access the DBMS.
2. Data structures consisting of fields, records, tables and files are optimized to handle large amounts of data stored on permanent data storage devices. This implies that the access to data will be slow compared to volatile main memory access.
3. The users interact with the database using a database query language and associated report writer. Depending on the privilege level, the user can read, write and update the records. Thus, a security system is also in place to protect the data from unauthorized access or tampering. The application programs may have to be customised to provide the various controls needed in a multi-user organization.
4. A transaction mechanism is in place to enable reliable processing of the data as desired by the user. Transaction is a single logical operation on the data incorporating ACID properties:
    a) Atomicity : to guarantee that either all tasks are performed on none. For instance, a fund transfer can be completed or it might fail. For instance, atomicity ensures that one account will not be credited if another is not debited as well.
    b) Consistency : to ensure the database rules and integrity constraints. If it is mandated that an account should have only positive balance, then no transaction that makes the balance negative will be allowed.
    c) Isolation : to make operations in a transaction appear isolated from all other operations. Outside the transaction, no operation can see the data in an intermediate state. Thus the transaction schedule is serializable while dealing with concurrent transactions. For performance reasons, this constraint may be relaxed, still ensuring the correctness of the transaction.
    d) Durability: Once the transaction is committed and user notified, the transaction will persist. The DBMS is expected to have checked the integrity constraints and the transaction will survive system failure. There will not be any need to abort the transaction.

However, it is not easy to implement the ACID properties. Based on speed requirement and storage considerations, techniques like write ahead logging, shadow paging etc., are used in conjuction with putting "locks" on the information acquired. Another method is to keep separate copies of data modified by different users. Networking and parallel processing environments need special handling mechanisms.

DBMSs form the core of database applications. It requests for data from the application program and instructs the operating system to transfer the appropriate data. They are sometimes built around a private multitasking kernel with built-in networking support. However, these functions are often left to the operating system on which they run. Database servers are specially designed computers that hold the actual databases and run only the DBMS and related software. Typically, they are multiprocessor computers using, for instance, disk arrays for stable storage.

DBMS handles "attributes" which are small chunks of information that describe some characteristic or property of a variable. Records are a collection of attributes while a table is a collection of records.

To start with, DBMS models used a "navigational" method to operate with linked lists of free-form records of data. With the increasing need for search facility, the relational model of database gained prominence where all data are split into a series of normalized tables. The database designer creates a consistent, logical representation of information using constraints, e.g. Foreign key, etc., declared as part of the logical schema. This is enforced by the DBMS for all applications.

The advent of object-oriented programming gave rise to ***object oriented database management system (ODBMS)***, where the database capabilities are combined with the object programming language capabilities. An ODBMS extends the programming language with transparently persistent data, concurrency control, data recovery, associative queries, and other capabilities. Some of the ODBMS work well with object-oriented programming languages such as Java, C#, Visual Basic, .NET, C++ and Smalltalk. Others have their own programming languages. ODBMS is useful in an enterprise when there is a need for high performance processing of complex data.

An **object-relational database** (**ORD**) or **object-relational database management system** (**ORDBMS**) is a relational database management system that allows developers to integrate the database with their own custom data types and methods. The term *object-relational database* is sometimes used to describe external software products running over traditional DBMSs to provide similar features. These systems are more correctly referred to as *object-relational* mapping systems.

4.2 PostgresSQL commands for data base operations

PostgreSQL falls into the category of *object-relational databases* having some object oriented features coupled with a firm relational database footing. It incorporates the concepts of classes, inheritance, types and functions, and its features include constraints, triggers, rules and transaction integrity.

Postgres DBMS originally began in 1986 and underwent many changes including changes in name to Postgres95, before settling down as PostgreSQL reflecting the SQL capability. In 1996, the version number was reset to start at 6.0 and the efforts were made to augement features and capabilities. The releaase 8.2 was towards the end of 2006. The current discussions relate to PostgreSQL versions which can be ported to any Unix/Posix-compatible system with full libc library support.

There are plethora of commands for managing databases in PostgreSQL such as for creating,

deleting or modifying the tables and databases; updating, inserting, removing, querying the tables; creating users, assigning permissions, etc. However, as this is not a DBMS text book, we are giving a brief idea of PostgreSQL commands so as to let the reader to use the PostgreSQL in an elementary way.

The various variable types supported by PostgreSQL are listed in table 4.1 given below.

Name	Aliases	Description
bigint	int8	signed eight-byte integer
bigserial	serial8	autoincrementing eight-byte integer
bit [(n)]		fixed-length bit string
bit varying [(n)]	varbit	variable-length bit string
boolean	bool	logical Boolean (true/false)
box		rectangular box in the plane
bytea		binary data ("byte array")
character varying [(n)]	varchar [(n)]	variable-length character string
character [(n)]	char [(n)]	fixed-length character string
cidr		IPv4 or IPv6 network address
circle		circle in the plane
date		calendar date (year, month, day)
double precision	float8	double precision floating-point number
inet		IPv4 or IPv6 host address
integer	int, int4	signed four-byte integer
interval [(p)]		time span
line		infinite line in the plane
lseg		line segment in the plane
macaddr		MAC address
money		currency amount
numeric [(p, s)]	decimal [(p, s)]	exact numeric of selectable precision
path		geometric path in the plane
point		geometric point in the plane
polygon		closed geometric path in the

Name	Aliases	Description
real	float4	single precision floating-point number
smallint	int2	signed two-byte integer
serial	serial4	autoincrementing four-byte integer
text		variable-length character string
time [(p)] [without time zone]		time of day
time [(p)] with time zone	timetz	time of day, including time zone
timestamp [(p)] [without time zone]		date and time
timestamp [(p)] with time zone	timestamptz	date and time, including time zone

Table 4.1 The variable type supported by PostgresSQL (source: PostgresSQL Manual)

Creating Tables

Now that we're in, we want to create a table. In the following example, *Create Table* command is used to create a table title "student" with attributes for Roll_No, Name, Telephone number and Date of Birth. The code segment will be as follows:

```
CREATE TABLE student (
“Roll_No”     int,
“Name”        varchar(30),
“Tel_No”      int,
“Date_of_Birth”      date);
```

Table 4.1 outlines the available attributes types.

We can also specify the primary key during the time of table creation. For example, in the following example Roll_No is specified as primary key.

```
CREATE TABLE student (
“Roll_No”     int PRIMARY KEY,
“Name”        varchar(30),
“Tel_No”      int,
“Date_of_Birth”      date);
```

We can even specify a combination of attributes as primary key such as follows:

```
CREATE TABLE student (
“Roll_No”     int,
“Name”        varchar(30),
“Tel_No”      int,
“Date_of_Birth”      date
PRIMARY KEY (Roll_No, Name)
);
```

The ALTER clause is used to add and remove column(s) to the table .

```
ALTER TABLE student ADD COLUMN city text;
ALTER TABLE student DROP COLUMN city;
```

Deleting Table

If we wanted to remove a table, we can use DROP command. For example:

DROP TABLE student;
will remove the table student.
To remove all rows from a table, we can run the following command. Remember that the table (meta structure) is still available.

DELETE FROM student;

Inserting Data
To insert data into the table, the INSERT command is used
.
INSERT INTO student VALUES (1,'Ramu', 256522, '2000-2-13');

Alternate way of inserting data is given below:
INSERT INTO student (Roll_No, Name, Tel_No, Date_of_Birth) VALUES (1,'Ramu', 256522, '2000-2-13');

A record can be inserted in the following manner.
INSERT INTO student (Roll_No, Tel_No, Date_of_Birth, Name) VALUES (1, 256522, '2000-2-13', 'Ramu');

To enter large amounts of data, we can use Tab-limited lines which could be cut and pasted from external applications. It is to be ensured that the data is not padded with spaces or commas. There should be just tabs, one for each column. One should remember to put "\." in the last line without fail.

COPY student FROM stdin;

2 Ravi 321212 2000-12-11

3 Anuh 212121 2005-12-1

\.

The data to be input could also be read from a file. However, the file should reside on the server only.

COPY student FROM /usr/local/data/mystudent.txt

Querying & Updating of Table Data

Updation is basically similar to insertion of data. In order to ensure correct update, the server side has to be told in which record(s) the data already exist and need to be changed. . This is done with the command "UPDATE" which has a "SET" clause to specify which fields to update and a "WHERE" clause to limit this update to a specific record (or multiple records). The WHERE clause allows one to carry the action on those rows or records for which the condition is true. The condition can involve relational operators (such as <, >, >=), logical operators (such as AND, OR, NOT).

In the following example, the Roll_No is set to 1 for those records whose Name value is "Ramu".

```
UPDATE student
SET Roll_No = 1
WHERE  Name = 'Ramu';
```

In the example given below, the Roll_No i set to 1 and Tel_No to 312121 for those records whose Name value is "Ramu".

```
UPDATE student
SET Roll_No = 1 ,
SET  Tel_No = 312121
WHERE  Name = 'Ramu';
```

It is possible to do arithmetic operations on the fields as given in the following example, where the Roll No is increased by one when the name is Ramu.

```
UPDATE student
SET Roll_No =  Roll_No + 1
WHERE  Name = 'Ramu';
```

To select all the rows of a table, we can run the following command :

```
SELECT * FROM  student;
```

To select specific columns, e.g., only Roll_No column of the table, we can run the following command :

```
SELECT Roll_NO FROM  student;
```

If we want the table content to be displayed in accordance with Date_of_Birth, the following command is run:

```
SELECT * FROM student ORDER BY  Date_of_Birth;
```

To display the table content (sorted or ordered) in accordance with Date_of_Birth and Tel_No, the following command is used :

```
SELECT * FROM student ORDER BY Date_of_Birth, Tel_No DESC;
```

If we want the Name column to be displayed in accordance with Date_of_Birth, we can run the following command.

```
SELECT Name FROM student ORDER BY Date_of_Birth;
```

If we want the table content to be displayed in accordance with Date_of_Birth and Tel_No, we can run the following command.

```
SELECT * FROM student GROUP BY date ORDER BY Tel_No DESC;
```

To eliminate duplicate records, we can use DISTINCT clause. For example, the following command displays names of the students, each name occuring only once. That is, if same name occurs more than once, there will not be multiple display.

```
SELECT DISTINCT Name FROM student;
```

Further, to sort the data, the command is as in the following example :

```
SELECT DISTINCT Name FROM student ORDER BY Tel_No;
```

We can carry out some operations on columns which are called as aggregate functions in DBMS terminology. For example, maximum of a column, min, average, standard deviation, etc., For example, the following example displays the largest Telephone number from our student table.

```
SELECT max(Tel_No) FROM student;
```

Assigning Privileges

In general, database owner can assign permissions to users at various levels. He can assign permissions on a whole database, views, table, row. The permissions can be for "read only" or modify as well. GRANT command is used for the purpose of managing privileges. One can refer

the manual pages of this command for variety of privileges available with PostgreSQL. For example, the following command grants UPDATE ability to the user on the database table student.

We can assign privileges to a group also. eg., we can give certain privileges to all using PUBLIC clause. We can use DROP, REVOKE clauses to manage privileges. The database owner (object owner) can assign or remove privileges at any time. Evidently, users can grant the privileges to other users.

JOIN Operations
In practice, we may need freedom to manage more than one table while answering a query. This is achieved by joining the tables. In DBMS literature, we have many types of joins like inner join, outer join, etc.

PostgreSQL supports variety of JOIN operations such as CROSS JOIN, INNER JOIN, LEFT INNER JOIN, RIGHT INNER JOIN, OUTER JOIN, FULL INNER JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN, etc.,.

For example, the following examples gives simple cross join of tables T1 and T2.

```
SELECT *  FROM T1 CROSS JOIN T2;
GRANT UPDATE on student to rao;
```

4.3 Server Management

Some useful commands related to the server management are discussed below.

To stop the PostgreSQL server:

```
pg_ctl  stop
```

The following can be used in recent versions.

```
/etc/rc.d/init.d/postgresql   stop
```

To start the Server

```
/etc/rc.d/init.d/postgresql start
```

Like most of the popular database systems, PostgreSQL also employs the client/server architecture (see Figure 4.1) . The server process is called as ‘postmaster’ and usual client processes can be pgsql (a front SQL interface) or even web browser/server. Figure 4.2 depicts how a query is executed in PostgreSQL.

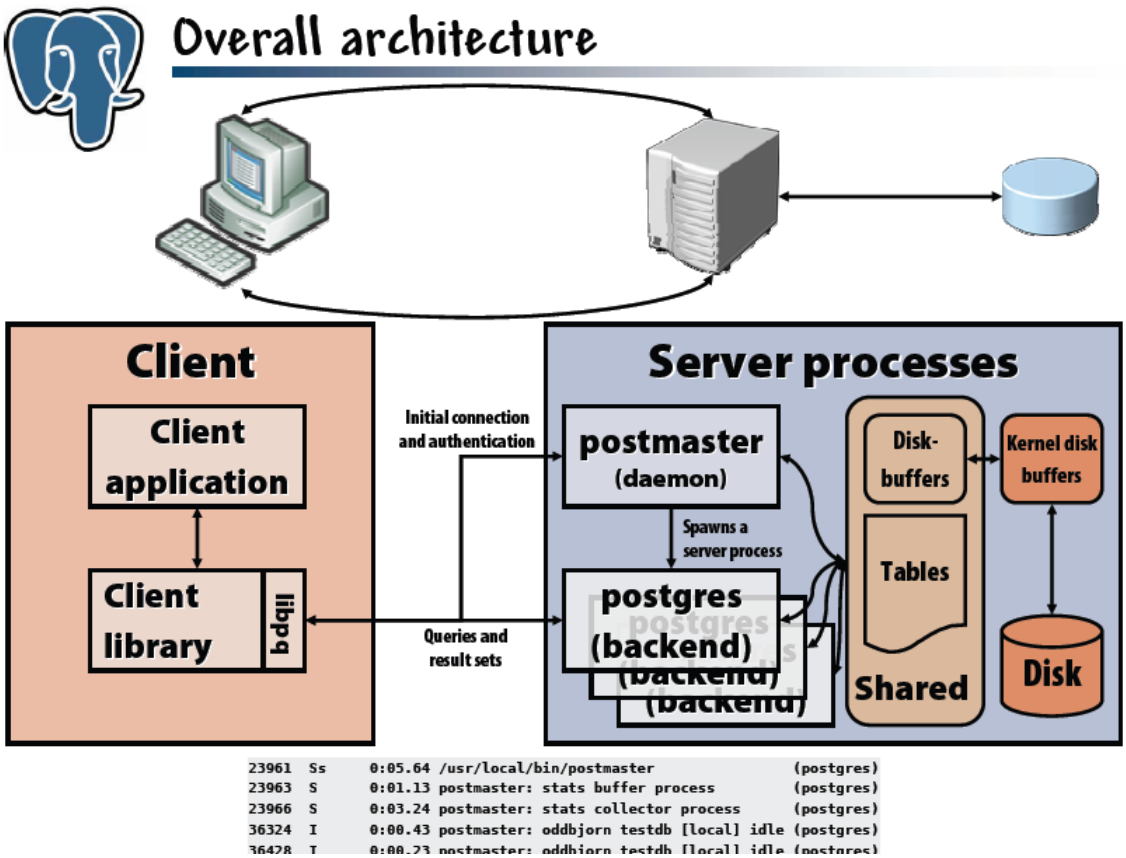

Figure 4.1 PostgreSQL Architecture

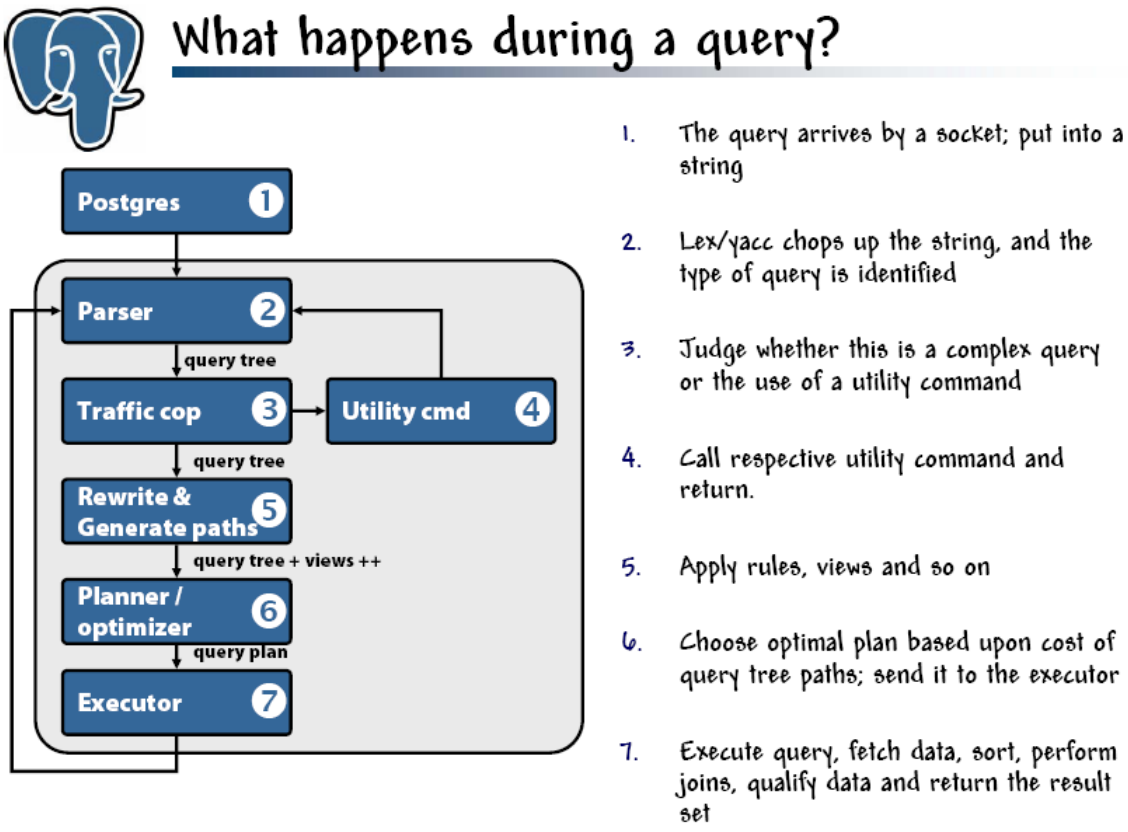

Figure 4.2 What happens when a query is executed

Probably, in a system default PostgreSQL server would have been installed. To check whether such

is the case, one could try to create a database with the command createdb as follows :

```
    createdb   exampledb
or
    /usr/local/pgsql/bin/createdb  exampledb
```

If there is an error message that 'createdb:command not found', then we can conclude that PostgreSQL is not installed or probably not configured. We can check up for the PostgreSQL tree under /usr/local/pgsql. If we find one, we can conclude that it is not configured though installed.

Also, do remember that PostgreSQL uses separate users list other than Unix/Linux users list. The server cannot be started like super user (root). First, we have to login as a normal user who is also a PostgreSQL user and then start the server.

Tools such as *vacuum* can be used to reclaim the free data blocks, update statistics. There exists pg_autovacuum to carry the same in an automatic manner.

Backup

For any database system, backup is very crucial. Every DBA has his or her own style of backup i.e., daily, hourly or weekly. Given below is a brief idea of the commands which can be used for backup purpose.

Very often, we may need a dump of a database system (text file), which represents the whole data base condition at that instant of time. To get this, we can use pg_dump command which normally outputs the database content on standard output. However, by redirecting the same into a file, we can have database content into a file which can be used either for hard copy production or restoring the database.

```
pg_dump exampledb
pg_dump exampledb > dumpfile
```

Please do remember that this command is having many other options. We are dealing in a minimalistic manner.

We can dump a database on a remote machine also with this command. To do this, we have to specify the host name with –h option. For example:

pg_dump –h ritchcenter exampledb >info

We can restore the database at a later stage with the help of this dump with the help of command such as the following.

psql dbname<info

We can also dump all the databases in a cluster and restore. For this purpose, we have to use pg_dumpall and psql.

pg_dumall >info

psql –f info postgres

Please do note that during restoring we have to use superuser userame such as 'postgres'.

We can also use tar command to backup the database. We know that the database is created in the directory such as /usr/local/pgsql/data. We can backup the database by giving this path to tar command. For example:

tar –cf backup.tar /usr/local/pgsql/data

For more details about tar options one can refer Venkateswarlu [2005].

However, this method has many drawbacks. For example, the database server has to be shut down before taking backup. Restoration is also little tricky. See the PostgreSQL manual pages for more details.

Similar to other databases, PostgreSQL also employs Write Ahead Logs (WAL) for the purpose of database recovery and consistency. All the transactions on the databases are maintained here. In the case of PostgreSQL, this is maintained as segments, each of which is 16MB. Unless otherwise mentioned, during the installation time (called as WAL archiving) the system recycles them to conserve the disk space. These segment files are best source of system backup and restoration.

Migration between releases

It is not uncommon to migrate from one version of database to another version. It is in the case of PostgreSQL users also. Always, there exists some differences between the releases. While migrating we have to be cautious with this point. We shall outline a brief step-by-step procedure for migrating to another release of the PostgreSQL.

It is wiser to use pg_dump or pg_dumpall commands to backup entire database system before migrating and maintain in the safe place.

It is possible that we can install new release of the PostgreSQL in another directory and run both old and new releases. Usually, the recent releases dump utilities can read the data files of previous releases. Thus, if we want we can let them to run on old databases and pipe the result to new PostgreSQL server in the following manner.

```
pg_dumpall   -p 5432 |psql –d postgres –p 3192
```

However, we have to run like superuser and both operations should run with different ports. On the other hand, if one does not want to run both the processes, the following steps could be taken :

1. On old server first run **pg_dumpall>backup**
2. stop old server by running **pg_ctl stop**
3. move old PostgreSQL tree to some other place by running command like **mv /usr/local/pgsql /usr/local/pgsql.old**.
4. download the new PostgreSQL server and extract the files.
5. Go to the directory such as postgresql-8.1.5
6. **qmake install**
7. **initdb –D /usr/local/pgsql/data**
8. **postmaster –D /usr/local/pgsql/data**
9. **psql –f backup postgres**

Database Replication

In some applications such as Data centers, we need multiple instances of our database for high

availability, backup or for a no-downtime migration to a new version. Slony-I is a trigger-based master to multiple slaves replication system for PostgreSQL being developed by Jan Wieck. This enterprise-level replication solution works asynchronously and offers all key features required by data centers. Among the key Slony-I usage scenarios are:

1. Database replication from the head office to various branches to reduce bandwidth usage or speed up database requests.
2. Database replication to offer load balancing in all instances. This can be particularly useful for report generators or dynamic Web sites.
3. Database replication to offer high availability of database services.
4. Hot backup using a standby server or upgrades to a new release of PostgreSQL.

Installing SLONY-I

To install Slony-I and replicate a simple database, first install PostgreSQL from source. Slony-I supports PostgreSQL 7.3.2 or higher; 7.4.x and 8.0 need the location of the PostgreSQL source tree when being compiled. If we prefer using PostgreSQL packages from our favorite distribution, simply rebuild them from the package sources and keep the package build location intact so it can be used when compiling Slony-I. That said, obtain the latest Slony-I release, compile and install it. To do so, proceed with the following commands:

```
% tar -zxvf slony1-1.0.5.tar.gz
% cd slony1-1.0.5
% ./configure \
--with-pgsourcetree=/usr/src/redhat/BUILD/postgresql-7.4.5
% make install
```

In this example, we tell the Slony-I's configure script to look in /usr/src/redhat/BUILD/postgresql-7.4.5/ for the location of the PostgreSQL sources, the directory used when building the PostgreSQL 7.4.5 RPMs on Red Hat Enterprise Linux. The last command compiles Slony-I and installs the following files:

- $postgresql_bindir/slonik: the administration and configuration script utility of Slony-I. slonik is a simple tool, usually embedded in shell scripts, used to modify Slony-I replication systems. It supports its own format-free command language described in detail in the Slonik Command Summary document.
- $postgresql_bindir/slon: the main replication engine. This multithreaded engine makes use of information from the replication schema to communicate with other engines, creating the distributed replication system.
- $postgresql_libdir/slony1_funcs.so: the C functions and triggers.
- $postgresql_libdir/xxid.so: additional datatype to store transaction IDs safely.
- $postgresql_datadir/slony1_base.sql: replication schema.
- $postgresql_datadir/slony1_base.v73.sql.
- $postgresql_datadir/slony1_base.v74.sql.
- $postgresql_datadir/slony1_funcs.sql: replication functions.
- $postgresql_datadir/slony1_funcs.v73.sql.

- $postgresql_datadir/slony1_funcs.v74.sql.
- $postgresql_datadir/xxid.v73.sql: a script used to load the additional datatype previously defined.

Generally, $postgresql_bindir points to /usr/bin/, $postgresql_libdir to /usr/lib/pgsql/ and $postgresql_datadir to /usr/share/pgsql/. Use the pg_config --configure command to display the parameters used when PostgreSQL was built to find the various locations for your own installation. Those files are all that one needs to offer a complete replication engine for PostgreSQL.

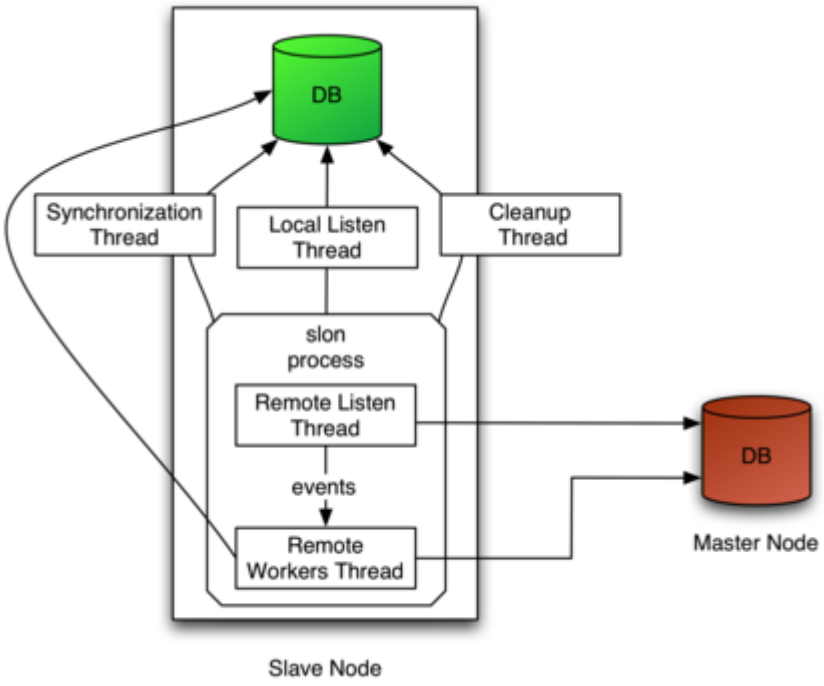

Figure 4.3. How the Slony-I replication engines work for a master with a slave database.

As can be see in Figure 4.3, Slony-I's main replication engine, Slon makes use of many threads. The synchronization thread verifies at a configurable interval if there has been replicable database activity, generating SYNC events if such activity happens. The local listen thread listens for new configuration events and modifies the cluster configuration and the in-memory configuration of the slon process accordingly.

As its name implies, the cleanup thread performs maintenance on the Slony-I schema, like removing old events or vacuuming the tables. The remote listen thread connects to the remote node's database to receive events from its event provider. When it receives events or confirmations, it selects the corresponding information and feeds the internal message queue of the remote workers thread. The replication data is combined into groups of transactions. The remote workers thread, one per remote node, does the actual data replication, events storing and generation of confirmations. At any moment, the slave knows exactly what groups of transactions it has consumed.

Replication of a Small Database

We first create the database we will replicate. This database contains a single table and sequence. Let's create a user contactuser, the contactdb database and activate the plpgsql programming language to this newly created PostgreSQL database by proceeding with the following commands:

```
% su - postgres
% createuser --pwprompt contactuser
Enter password for user "contactuser": (specify a
password)
Enter it again:
Shall the new user be allowed to create databases?
(y/ n) y
Shall the new user be allowed to create more new
users? (y/ n) n
% createdb -O contactuser contactdb
% createlang -U postgres -h localhost plpgsql \
contactdb
```

Then, we create the sequence and the table in the database we will replicate and insert some information in the table:

```
% psql -U contactuser contactdb
contactdb=> create sequence contact_seq start with 1;
contactdb=> create table contact (
 cid         int4 primary key,
 name        varchar(50),
 address     varchar(255),
 phonenumber varchar(15)
);
contactdb=> insert into contact (cid, name, address,
phonenumber) values ((select nextval('contact_seq')),
'Joe', '1 Foo Street', '(592) 471-8271');
contactdb=> insert into contact (cid, name, address,
phonenumber) values ((select nextval('contact_seq')),
'Robert', '4 Bar Roard', '(515) 821-3831');
contactdb=> \q
```

For the sake of simplicity, let's create a second database on the same system in which we will replicate the information from the contactdb database. Proceed with the following commands to create the database, add plpgsql programming language support and import the schema without any data from the contactdb `database:`

```
% su - postgres
% createdb -O contactuser contactdb_slave
% createlang -U postgres -h localhost plpgsql \
contactdb_slave
% pg_dump -s -U postgres -h localhost contactdb | \
psql -U postgres -h localhost contactdb_slave
```

Once the databases are created, we are ready to create our database cluster containing a master and a single slave. Create the Slonik cluster_setup.sh script and execute it. Listing 1 shows the content of the cluster_setup.sh script.

Listing 1. cluster_setup.sh

```
#!/bin/sh
CLUSTER=sql_cluster
```

```
DB1=contactdb
DB2=contactdb_slave
H1=localhost
H2=localhost
U=postgres
slonik <<_EOF_
cluster name = $CLUSTER;
node 1 admin conninfo = 'dbname=$DB1 host=$H1 user=$U';
node 2 admin conninfo = 'dbname=$DB2 host=$H2 user=$U';
init cluster (id = 1, comment = 'Node 1');
create set (id = 1, origin = 1,
      comment = 'contact table');
set add table (set id = 1, origin = 1, id = 1,
       full qualified name = 'public.contact',
       comment = 'Table contact');
set add sequence (set id = 1, origin = 1, id = 2,
       full qualified name = 'public.contact_seq',
       comment = 'Sequence contact_seq');
store node (id = 2, comment = 'Node 2');
store path (server = 1, client = 2,
     conninfo = 'dbname=$DB1 host=$H1 user=$U');
store path (server = 2, client = 1,
         conninfo = 'dbname=$DB2 host=$H2 user=$U');
store listen (origin = 1, provider = 1, receiver = 2);
store listen (origin = 2, provider = 2, receiver = 1);
```

The first slonik command (cluster name) of Listing 1 defines the namespace where all Slony-I-specific functions, procedures, tables and sequences are defined. In Slony-I, a node is a collection of a database and a slon process, and a cluster is a collection of nodes, connected using paths between each other. Then, the connection information for node 1 and 2 is specified, and the first node is initialized (init cluster). Once completed, the script creates a new set to replicate, which is essentially a collection containing the public.contact table and the public.contact_seq sequence. After the creation of the set, the script adds the contact table to it and the contact_seq sequence. The store node command is used to initialize the second node (id = 2) and add it to the cluster (sql_cluster). Once completed, the scripts define how the replication system of node 2 connects to node 1 and how node 1 connects to node 2. Finally, the script tells both nodes to listen for events (store listen) for every other node in the system.

Once the script has been executed, start the slon replication processes. A slon process is needed on the master and slave nodes. For our example, we start the two required processes on the same system. The slon processes must always be running in order for the replication to take place. If for some reason they must be stopped, simply restarting allows them to continue where they left off. To start the replication engines, proceed with the following commands:

```
% slon sql_cluster "dbname=contactdb user=postgres" &
% slon sql_cluster "dbname=contactdb_slave user=postgres" &
```

Next, we need to subscribe to the newly created set. Subscribing to the set causes the second node, the subscriber, to start replicating the information of the contact table and contact_seq sequence from the first node. Listing 2 shows the content of the subscription script.

Listing 2. subscribe.sh

```
#!/bin/sh
CLUSTER=sql_cluster
```

```
DB1=contactdb
DB2=contactdb_slave
H1=localhost
H2=localhost
U=postgres
slonik <<_EOF_
cluster name = $CLUSTER;
node 1 admin conninfo = 'dbname=$DB1 host=$H1 user=$U';
node 2 admin conninfo = 'dbname=$DB2 host=$H2 user=$U';
subscribe set (id = 1, provider = 1, receiver = 2, forward = yes);
```

Much like Listing 1, subscribe.sh starts by defining the cluster namespace and the connection information for the two nodes. Once completed, the subscribe set command causes the first node to start replicating the set containing a single table and sequence to the second node using the slon processes.

Once the subscribe.sh script has been executed, connect to the contactdb_slave database and examine the content of the contact table. At any moment, you should see that the information was replicated correctly:

```
% psql -U contactuser contactdb_slave
contactdb_slave=> select * from contact;
 cid |  name  |   address    |  phonenumber
-----+--------+--------------+----------------
   1 | Joe    | 1 Foo Street | (592) 471-8271
   2 | Robert | 4 Bar Roard  | (515) 821-3831
```

Now, connect to the /contactdb/ database and insert a row:

```
% psql -U contact contactdb
contactdb=> begin; insert into contact (cid, name,
 address, phonenumber) values
 ((select nextval('contact_seq')), 'William',
 '81 Zot Street', '(918) 817-6381'); commit;
```

Examining the contents of the contact table of the contactdb_slave database once more, one will notice that the row was replicated. Now, to delete a row from the /contactdb/ database:

```
contactdb=> begin; delete from contact
 where cid = 2; commit;
```

Again, by examining the content of the contact table of the contactdb_slave database, it can be noticed that the row was removed from the slave node correctly.

Instead of comparing the information for contactdb and contactdb_slave manually, we easily can automate this process with a simple script, as shown in Listing 3. Such a script could be executed regularly to ensure that all nodes are in sync, notifying the administrator if that is no longer the case.

Listing 3. compare.sh

```
#!/bin/sh
CLUSTER=sql_cluster
```

```
DB1=contactdb
DB2=contactdb_slave
H1=localhost
H2=localhost
U=postgres
echo -n "Comparing the databases..."
psql -U $U -h $H1 $DB1 >dump.tmp.1.$$ <<_EOF_
        select 'contact'::text, cid, name, address,
          phonenumber from contact order by cid;
_EOF_
psql -U $U -h $H2 $DB2 >dump.tmp.2.$$ <<_EOF_
        select 'contact'::text, cid, name, address,
          phonenumber from contact order by cid;
_EOF_
if diff dump.tmp.1.$$ dump.tmp.2.$$ >dump.diff ; then
        echo -e "\nSuccess! Databases are identical."
        rm dump.diff
else
        echo -e "\nFAILED - see dump.diff."
fi
rm dump.tmp.?.$$
```

Although replicating a database on the same system isn't of much use, this example shows how easy it is to do. If you want to experiment with a replication system on nodes located on separate computers, you simply would modify the DB2, H1 and H2 environment variables from Listing 1 to 3. Normally, DB2 would be set to the same value as DB1, so an application always refers to the same database name. The host environment variables would need to be set to the fully qualified domain name of the two nodes. You also would need to make sure that the slon processes are running on both computers. Finally, it is good practice to synchronize the clocks of all nodes using ntpd or something similar.

Later, if you want to add more tables or sequences to the initial replication set, you can create a new set and use the merge set slonik command. Alternatively, you can use the set move table and set move sequence commands to split the set. Refer to the Slonik Command summary for more information on this.

In case of a failure from the master node, due to an operating system crash or hardware problem, for example, Slony-I does not provide any automatic capability to promote a slave node to become a master. This is problematic because human intervention is required to promote a node, and applications demanding highly available database services should not depend on this. Luckily, plenty of solutions are available that can be combined with Slony-I to offer automatic failover capabilities. The Linux-HA Heartbeat program is one of them.

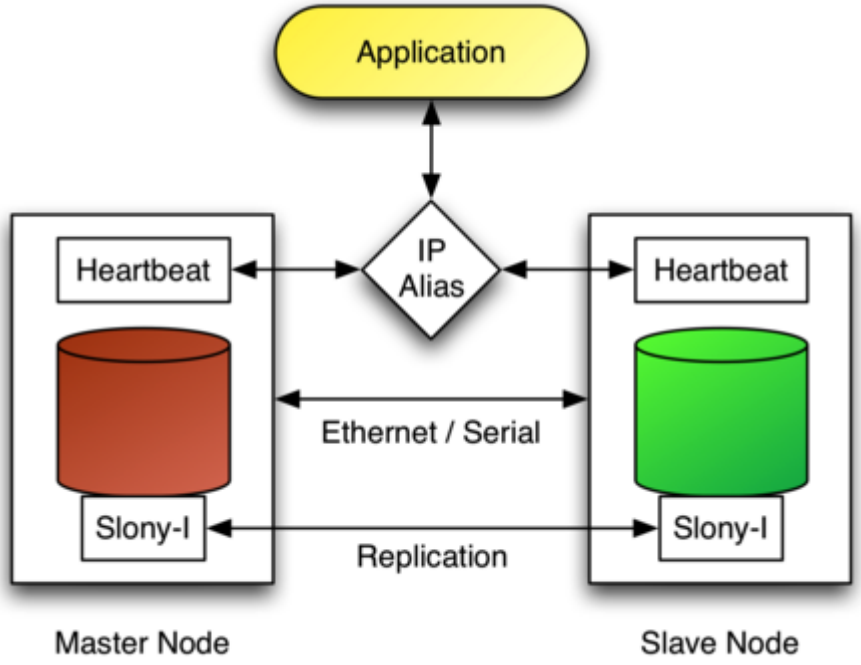

Figure 4.4. Heartbeat switches the IP alias to the slave node in case the master fails.

Consider Figure 4.4, which shows a master and slave node connected together using an Ethernet and serial link. In this configuration, the Heartbeat is used to monitor the node's availability through those two links. The application makes use of the database services by connecting to PostgreSQL through an IP alias, which is activated on the master node by the Heartbeat. If the Heartbeat detects that the master node has failed, it brings the IP alias up on the slave node and executes the slonik script to promote the slave as the new master.

The script is relatively simple. Listing 4 shows the content of the script that would be used to promote a slave node, running on slave.example.com, so it starts offering all the database services that master.example.com offered.

Listing 4. promote.sh

```
#!/bin/bash
CLUSTER=sql_cluster
H1=master.example.com
H2=slave.example.com
U=postgres
DB1=contactdb
DB2=contactdb
su - postgres -c slonik <<_EOF_
cluster name = $CLUSTER;
node 1 admin conninfo = 'dbname=$DB1 host=$H1 user=$U';
node 2 admin conninfo = 'dbname=$DB2 host=$H2 user=$U';
failover (id = 1, backup node = 2);
drop node (id = 1, event node = 2);
```

From Listing 4, the failover Slonik command is used to indicate that the node with id = 1, the node running on master.example.com, has failed, and that the node with id = 2 will take over all sets from the failed node. The second command, drop node, is used to remove the node with id = 1 from

the replication system completely. Eventually, you might want to bring back the failed node in the cluster. To do this, you must configure it as a slave and let Slony-I replicate any missing information. Eventually, you can proceed with a switchback to the initial master node by locking the set (lock set), waiting for all events to complete (wait for event), moving the set to a new origin (move set) and waiting for a confirmation that the last command has completed. Please refer to the Slonik Command Summary for more information on those commands.

Other replication servers are given in Figure 4.5

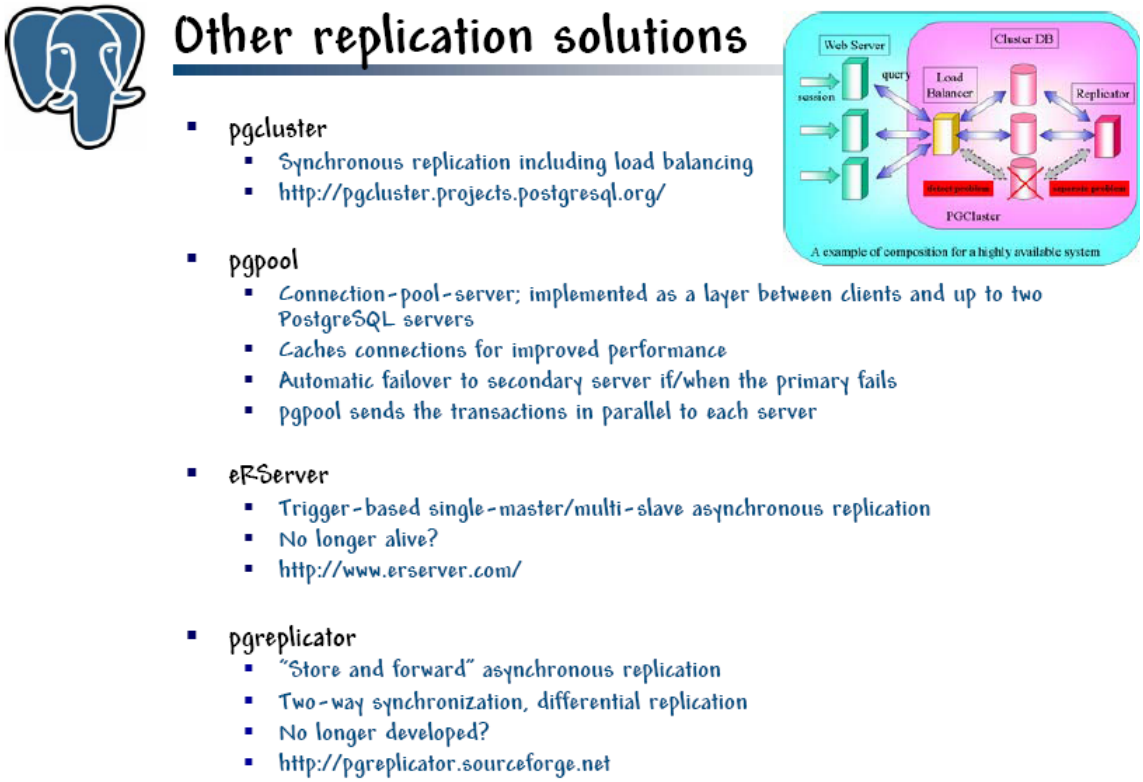

Figure 4.5 Other replication servers for PostgreSQL

4.4 Introduction to PgAdmin, a GUI based Server Manager

The PgAdmin is a GUI based tool for administering PostgreSQL server. It can be downloaded from www.pgadmin.org. Currently it supports PostgreSQL release 8 and above. With its help, we can carry out many administration tasks easily.

Some screenshots are given below (Figure 4.6, 4.7).

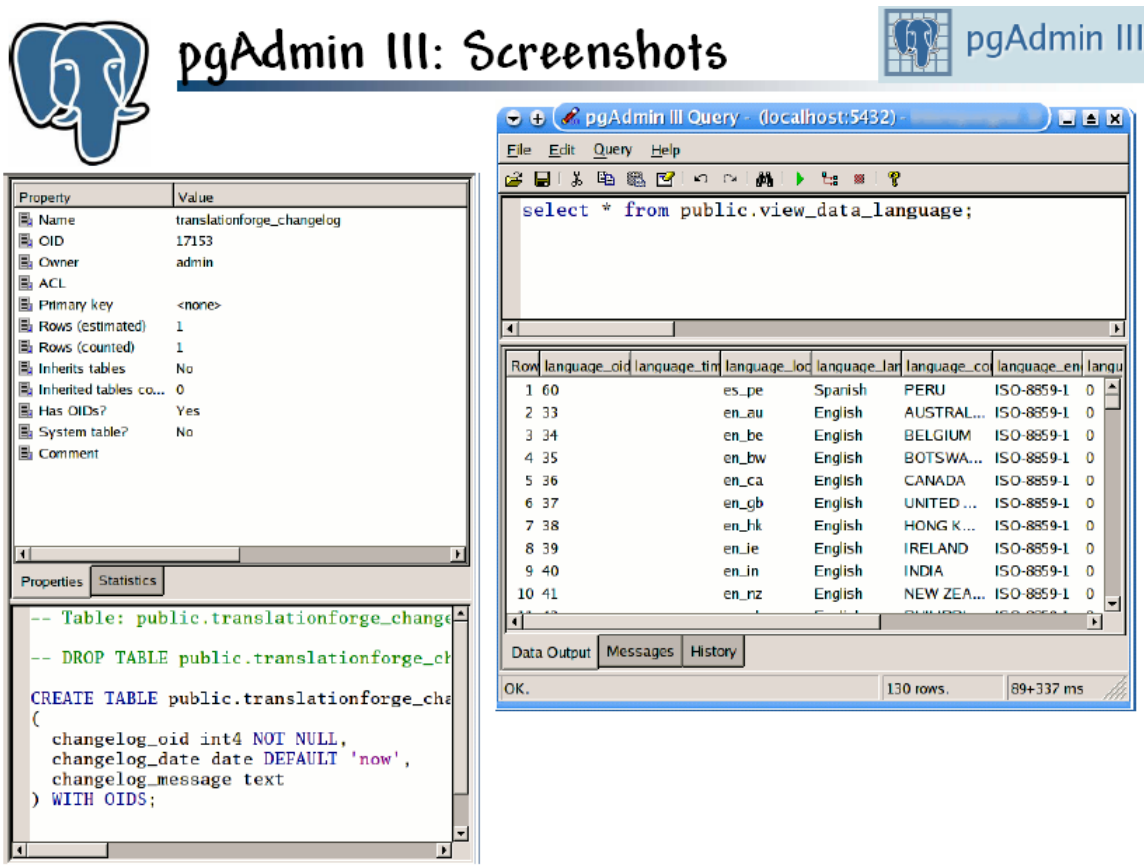

Figure 4.6 A sample screenshot of PgAdmin

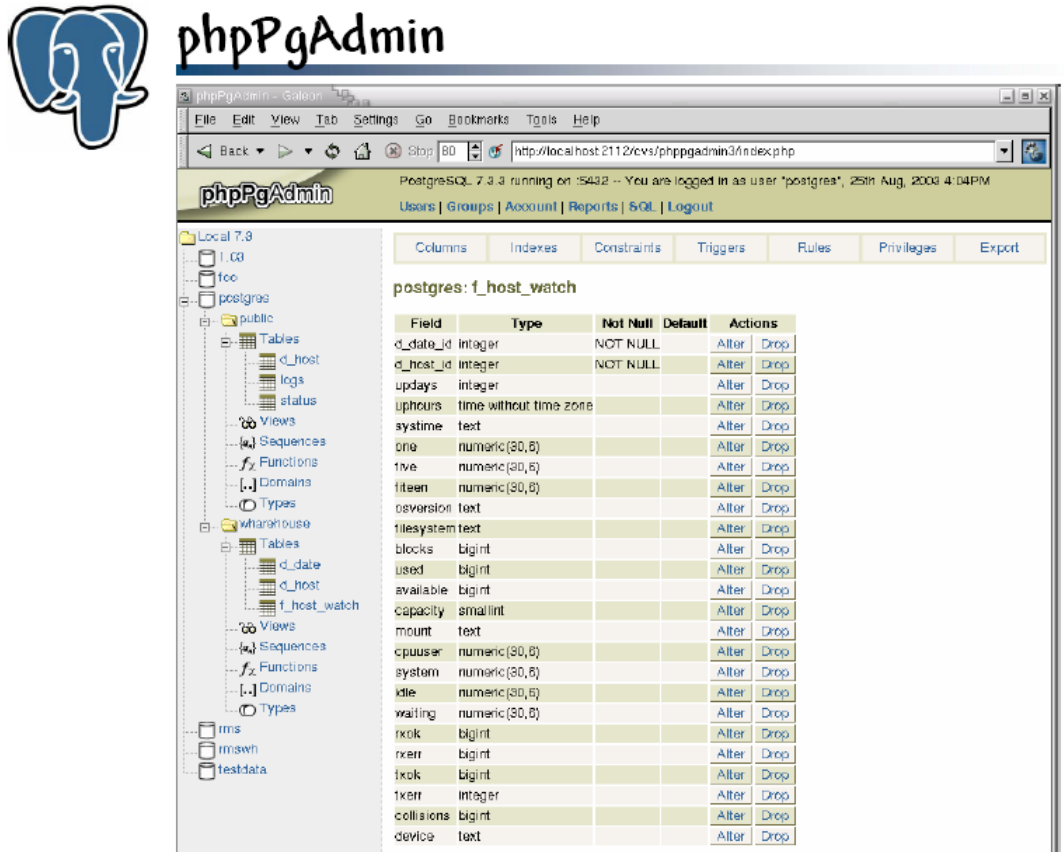

Figure 4.7 A screenshot of PgAdmin giving details of the table metadata

4.5 Web enabled databases and need for Metadata standards: Dublin Core – A Meta Data Standard

Now a days, every organization is making its presence visible in the Internet in contrast to having a simple web pages in the earlier days. All of its information sources are made available through Internet. Usually, the databases are developed under Oracle, MySQL, PostgreSQL, etc., and made accessed through web pages either with Java Script, JSP, perl, python, cgi,ruby,Java Servlets, ASP scripts. While doing so, we may often use metadata of the tables (metadata is the data about the data!!).

4.5.1. What is Metadata?

Metadata is structured data which describes the characteristics of a resource. It shares many similar characteristics to the cataloguing that takes place in libraries, museums and archives. The term "meta" derives from the Greek word denoting a nature of a higher order or more fundamental kind. A metadata record consists of a number of pre-defined elements representing specific attributes of a resource, and each element can have one or more values. Table 4.2 given below is an example of a simple metadata record.

Element name	Value
Title	Web catalogue
Creator	Dagnija McAuliffe
Publisher	University of Queensland Library
Identifier	http://www.library.uq.edu.au/iad/mainmenu.html
Format	Text/html
Relation	Library Web site

Table 4.2 : A sample metadata record

Each metadata schema will usually have the following characteristics:

- a limited number of elements
- the name of each element
- the meaning of each element

Typically, the semantics is descriptive of the contents, location, physical attributes, type (e.g. text or image, map or model) and form (e.g. print copy, electronic file). Key metadata elements supporting access to published documents include the originator of a work, its title, when and where it was published and the subject areas it covers. Where the information is issued in analog form, such as print material, additional metadata is provided to assist in the location of the information, e.g. call numbers used in libraries. The resource community may also define some logical grouping of the elements or leave it to the encoding scheme. For example, Dublin Core may provide the core to which extensions may be added.

Some of the most popular metadata schemas include:

- Dublin Core
- AACR2 (Anglo-American Cataloging Rules)
- GILS (Government Information Locator Service)
- EAD (Encoded Archives Description)
- IMS (IMS Global Learning Consortium)

- AGLS (Australian Government Locator Service)

While the syntax is not strictly part of the metadata schema, the data will be unusable, unless the encoding scheme understands the semantics of the metadata schema. The encoding allows the metadata to be processed by a computer program. Important schemes include:

- HTML (Hyper-Text Markup Language)
- SGML (Standard Generalised Markup Language)
- XML (eXtensible Markup Language)
- RDF (Resource Description Framework)
- MARC (MAchine Readable Cataloging)
- MIME (Multipurpose Internet Mail Extensions)

Metadata may be deployed in a number of ways:

- Embedding the metadata in the Web page by the creator or their agent using META tags in the HTML coding of the page
- As a separate HTML document linked to the resource it describes
- In a database linked to the resource. The records may either have been directly created within the database or extracted from another source, such as Web pages.

The simplest method is for Web page creators to add the metadata as part of creating the page. Creating metadata directly in a database and linking it to the resource, is growing in popularity as an independent activity to the creation of the resources themselves. Increasingly, it is being created by an agent or third party, particularly to develop subject-based gateways.

The search engines uses the metadata while searching for the resource.

A more formal definition of metadata offers a clue:

> *Metadata is data associated with objects which relieves their potential users of having full advance knowledge of their existence or characteristics.*

Information resources must be made visible in a way that allows people to tell whether the resources are likely to be useful to them. This is no less important in the online world, and in particular, the World Wide Web. Metadata is a systematic method for describing resources and thereby improving access to them. If a resource is worth making available, then it is worth describing it with metadata, so as to maximise the ability to locate it.

Metadata provides the essential link between the information creator and the information user. While the primary aim of metadata is to improve resource discovery, metadata sets are also being developed for other reasons, including:

- administrative control
- security
- personal information
- management information
- content rating
- rights management
- preservation

While the current discussion focuses on resource discovery and retrieval, the above additional purposes for metadata should also be kept in mind.

4.5.2 Which Metadata schema?

There are literally hundreds of metadata schemas to choose from and the number is growing rapidly, as different communities seek to meet the specific needs of their members. Recognising the need to answer the question of how can a simple metadata record be defined that sufficiently describes a wide range of electronic documents, the Online Computer Library Center (OCLC) of which the University of Queensland Library is currently the only full member in Australia, combined with the National Center for Supercomputing Applications (NCSA) to sponsor the first Metadata Workshop in March, 1995 in Dublin, Ohio [DC1]. The primary outcome of the workshop was a set of 13 elements (subsequently increased to 15) named the Dublin Metadata Core Element Set (known as Dublin Core). Dublin Core was proposed as the minimum number of metadata elements required to facilitate the discovery of document-like objects in a networked environment such as the Internet.

Table 4.3 below is a summary of the elements in Dublin Core. The metadata elements fall into three groups which roughly indicate the class or scope of information stored in them: (1) elements related mainly to the content of the resource, (2) elements related mainly to the resource when viewed as intellectual property, and (3) elements related mainly to the physical manifestation of the resource.

Content & about the Resource	**Intellectual Property**	**Electronic or Physical manifestation**
Title	Author or Creator	Date
Subject	Publisher	Type
Description	Contributor	Format
Source	Rights	Identifier
Language		
Relation		
Coverage		

Table 4.3 Summary elements in Dublic Core

Given below is an example of a Dublin Core record for a short poem, encoded as part of a Web page using the <META> tag:

```
<HTML> !4.0!
<HEAD>
<TITLE>Song of the Open Road</TITLE>
<META NAME="DC.Title" CONTENT="Song of the Open Road">
<META NAME="DC.Creator" CONTENT="Nash, Ogden">
<META NAME="DC.Type" CONTENT="text">
<META NAME="DC.Date" CONTENT="1939">
<META NAME="DC.Format" CONTENT="text/html">
<META NAME="DC.Identifier" CONTENT="http://www.poetry.com/nash/open.html">
</HEAD>
<BODY><PRE>
I think that I shall never see
A billboard lovely as a tree.
Indeed, unless the billboards fall
```

I'll never see a tree at all.
</PRE></BODY>
</HTML>

The <META> tag is not normally displayed by Web browsers, but can be viewed by selecting "Page Source".

In addition to the 15 elements, three qualifying aspects have been accepted to enable the Dublin Core to function in an international context and also meet higher level scientific and subject-specific resource discovery needs.

These three Dublin Core Qualifiers are:

- LANG: indicating the language of the contents of the element, to be used in both resource discovery and in filtering retrieval results
- SCHEME: indicating the set of regulations, standards, conventions or norms from which a term in the content of the element has been taken
- SUB-ELEMENT: refinement of some of the elements to gain more precision

4.5.3 Why Dublin Core?

The Dublin Core metadata schema offers the following advantages:

- Its useability and its flexibility
- The semantics of these elements is designed to be clear enough to be understood by a wide range of customers, without the need for training
- The elements of Dublin Core are easily identifiable by having the work in hand, such as intellectual content and physical format
- It is not intended to supplant other resource descriptions, but rather to complement them. It is intended to describe the essential features of electronic documents that support resource discovery. Other important metadata such as accounting and archival data, were deliberately excluded to keep the schema as simple and useable as possible.
- It is mostly syntax independent, to support its use in the widest range of applications
- All elements are optional, but allows each site to define which elements are mandatory and which are optional
- All elements are repeatable
- The elements may be modified in limited and well-defined ways through the use of specific qualifiers, such as the name of the thesaurus used in the subject element
- It can be extended to meet the demands of more specialised communities. From the very beginning, the Dublin Core creators recognised that some resources could not be adequately described by a small set of elements. The Dublin Core creators came up with two solutions. Firstly, by allowing the addition of elements for site-specific purposes or specialised fields. Secondly, by designing the Dublin Core schema so that it could be mapped into more complex and tightly controlled systems, such as MARC.

Dublin Core has received widespread acceptance amongst the resource discovery community and has become the de facto Internet metadata standard [AGLS, p.3].
To date, the depth of implementation in individual sectors has been patchy. In Australia, much activity has taken place in the government sector, under the auspices of the Government Technology and Telecommunications Committee (GTTC). Dublin Core has been formally accepted as the standard for the Australian Government Locator Service [AGLS].

4.5.4 Where will the metadata be stored?

Metadata may be deployed in a number of ways:

- Embedding the metadata in the Web page by the creator or their agent using META tags in the HTML coding of the page
- As a separate HTML document linked to the resource it describes

- In a database linked to the resource. The records may either have been directly created within the database or extracted from another source, such as Web pages.

The simplest method is to ask Web page creators to add the metadata as part of creating the page. To support rapid retrieval, the metadata should be harvested on a regular basis by the site robot. This is currently by far the most popular method for deploying Dublin Core. An increasing range of software is being made available to assist in the addition of metadata to Web pages.

Creating metadata directly in a database and linking it to the resource, is growing in popularity as an independent activity to the creation of the resources themselves. Increasingly, it is being created by an agent or third party, particularly to develop subject-based gateways. The University of Queensland Library is involved in a number of gateway projects, including AVEL and Weblaw.

4.5.5 How does one create metadata?

The more easily the metadata can be created and collected at point of creation of a resource or at point of publication, the more efficient the process and the more likely it is to take place. There are many such tools available and the number continues to grow. Such tools can be standalone or part of a package of software, usually with a backend database or repository to store and retrieve the metadata records, Some examples include:

- DC-dot - http://www.ukoln.ac.uk/metadata/dcdot/. This service will retrieve a Web page and automatically generate Dublin Core metadata, either as HTML tags or as RDF/XML, suitable for embedding in the section of the page.
- DCmeta - http://www.dstc.edu.au/RDU/MetaWeb/generic_tool.html. Developed by Tasmania Online. It is based on SuperNoteTab text-editor and can be customised.
- HotMeta - http://www.dstc.edu.au/Research/Projects/hotmeta/. A package of software, including metadata editor, repository and search engine.

Ideally, metadata should be created using a purpose-built tool, with the manual creation of data kept to an absolute minimum. The tool should support:

- Inclusion of the syntax in the template (e.g. element name, sub-element, qualifier)
- Default content, which can be overridden
- Content selected from a list of limited choices (e.g. Function, Type, Format)
- Validation of mandatory elements, sub-elements, schemes and element values

In this chapter, the middle layer where all the data gets stored is described. After a brief overview of database management system with PostgreSQL as example, the server management is discussed introducing PgAdmin, a GUI based server manager tool. The final section deals with web enabled database and the Dublin Core, a meta data standard.

Chapter 5

Business Logic Layer : Evolution of Enterprise Applications

Any business application can be divided into three parts based on the functions performed:

- Presentation Logic -- presentation of the objects that embodies the system to the user.
- Business Logic -- business logic of a software system (sometimes called the domain logic because it relates to the problem domain) that deals with the performance of business-related tasks, i.e., focussing on the processing of the data
- Datastore Logic -- Part of the system where all the data are stored.

Presentation and Datastore aspects were discussed in the previous chapters. In the current chapter, we will study Business logic which refers to the business rules in a problem domain rather than the view of data or storage. Business Logic Layer provides services to the presentation layer and manipulates the Data layer using appropriate data access logic. It can also make requests to external services. This layer deals with the services offered, the application logic and rules to achieve the desired functionality, and the server-side operations.

The applications could be written as a monolith structure or structured into different layers wherein the software architecture describes how the different parts are handled. Application (software) architectural patterns are used to provide well-established solutions to software engineering architectural problems. The pattern is essentially fundamental structural organization schema for a software system, consisting of predefined subsystems and specify their responsibilities and relations.

Just as in the case of the earlier chapters related to presentation and datastore logic, the features of the business logic layer will be discussed under the MVC paradigm.

5.1 Application Architectures

Application architectures have evolved over the course of time from 1-tier architecture to 2-tier, 3-tier and N-tier architecture. We also have patterns like Model-View-Controller (MVC) concepts, Client-Server, service-oriented architecture, pipeline, implicit invocation, etc. The architecture reflects the flexibility needed to make the system and where they are being used. We shall take a brief look into some of the architectures.

1-tier application

In this case, data access, presentation and business logic, all are tightly coupled in one monolithic application 1-Tier Architecture. This represents the oldest type of business application.

Advantages

1. easy to optimize performance
2. no context switching
3. no compatibility issues
4. no client development, maintenance and deployment cost
5. data consistency is easy to maintain

Disadvantages

1. high maintenance cost
2. hard to modify
3. code reuse difficult

2 Tier (Client/Server architecture)

In the 2-tier architecture, presentation, business and data model processing logic are present in the client application. Server is typically a database server. Client sends SQL statements, retrieves raw data from the database server and processes and displays the results. Here the processing is visible to the client. This architecture became prominent once networking computers became cheap enough for businesses.

While the main advantage is the database product independence, the disadvantages and issues include the following :

1. Deployment costs are high
2. Database switching costs are high
3. Achieving high scalability is difficult
4. Performance is not very high
5. Security issues are difficult to solve
6. Code reuse is difficult

Distributed Object architecture [3-tier]

This type of architecture was developed during late 80s to early 90s. In this model, the user interface, business logic and data model (storage and access) are developed and maintained as independent modules, as independent distributed objects (solutions available from Java RMI-IIOP, CORBA, Microsoft .NET). All communications between the various objects are done using object request protocols like IIOP, RMI, DCOM/COM+. The middle tier handles concurrency, multi-threading, connection pooling etc. This architecture has the usual advantages of modular software with well defined interface and each of the three layers could be independently upgraded or replaced depending on the requirement or on technology change. For instance, if the operating system is changed from Microsoft Windows to Linux, only the user interface code gets affected.

This architecture, in spite of providing much of the functionality in the platform itself, is often difficult to program

Adding Web tier [N-tier]

Once the World Wide Web became popular, new types of applications had to be developed. The original technology use for writing Web applications, CGI/Perl is difficult to maintain, and had severe scalability issues. Nowadays a mixture of different technologies are used.

Moving from 1-tier to n-tier resulted in a distinct separation of responsibilities. Software was developed such that, change in a specific tier was possible without disturbing other tiers resulting in easier maintenance and deployment. N-tier development also proved to be reusable, flexible and scaleable. However, it also requires developer to worry about the details of distributed protocol, security, load balancing, concurrency, transactions, resource management, etc. To alleviate this tedium, people started using Application servers which provided most of the functionality and solved most of the tedious problems for the developers.

5.2 Model-View-Controller (MVC) Architecture

Model-view-controller (MVC) is a software architecture that separates an application's data model, user interface, and control logic into three distinct components so that modifications to one component can be made with minimal impact to the others. MVC is often thought of as a software design pattern. However, MVC encompasses more of the architecture of an application than is typical for a design pattern. Hence the term architectural pattern is appropriate. MVC pattern is basically layered architecture for a Web application. It evolves from Front Controller pattern.

The Front Controller pattern defines a single component that is responsible for processing application requests, centralizes functions such as view selection, security, and templating, and applies them consistently across all pages or views. When the behavior of these functions need to change, only a small part of the application needs to be changed: the controller and its helper.

Model-View-Controller architecture is often used for developing web-applications. This model is for decoupling between business logic and data presentation to web user. This model divides the web based application into three layers (Fig. 5.1):

Model:
Model contains the business logics and functions that manipulate the application data. It provides updated information to view domain and the controller can access the functionality which is encapsulated in the model.

View:
View is responsible for presentation aspect of application according to the current state of model data and is also responsible to query and response to the controller.

Controller:
Controller accepts and intercepts user requests and controls the business objects to fulfill these requests.

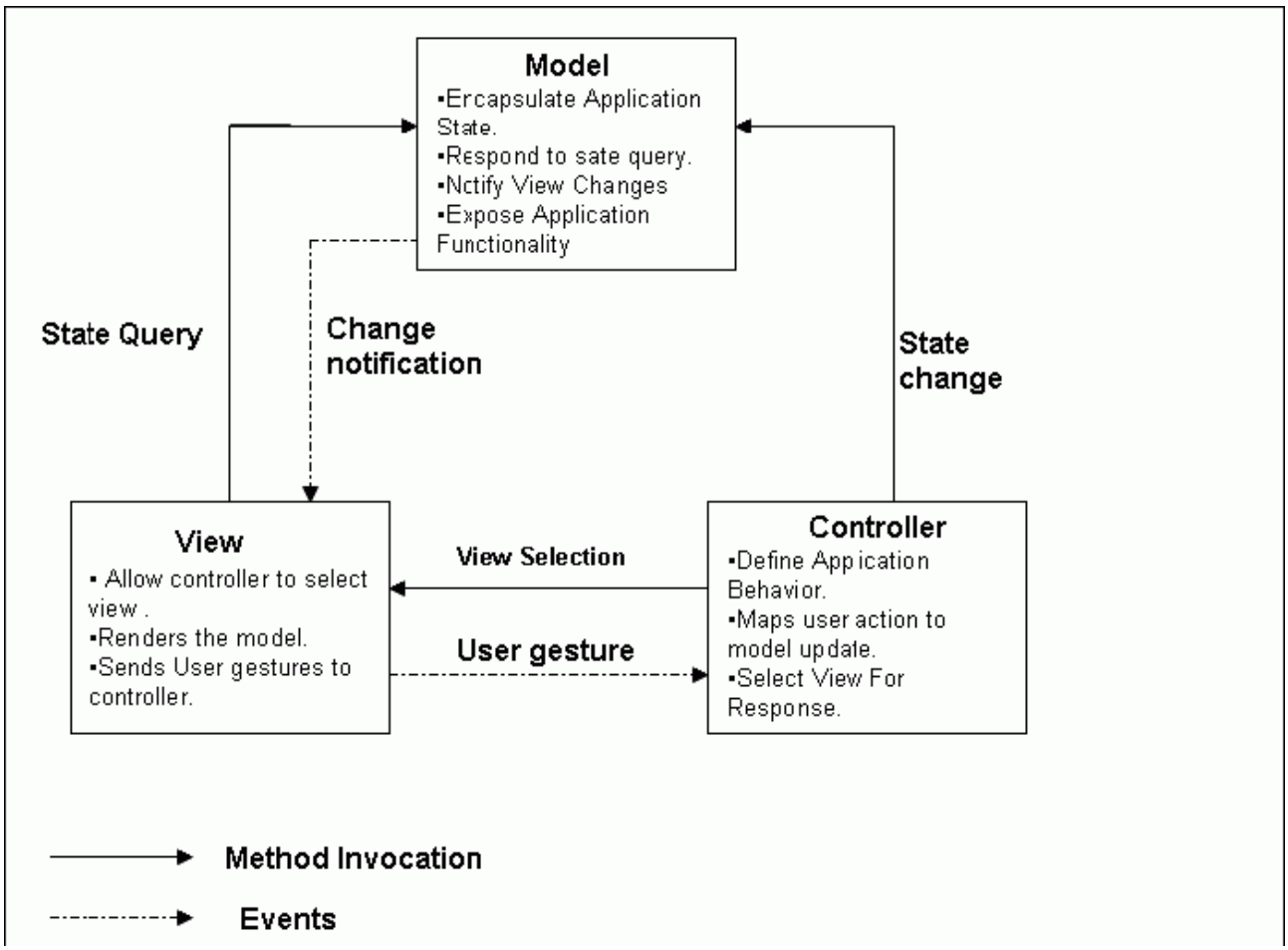

Fig. 5.1 Model-View-Control Architecture

The Model Layer is designed for capturing logic and data access code. It is a self contained layer and can function independently from the view and control layer. Mostly implemented using technologies like Enterprise JavaBeans, Java Data Objects, Data Access Objects etc. This layer will be the focus on the current chapter.

The View layer is designed for providing an interface to application for users. It is the conduit for getting data in and out of the application. It does not contain any code for persisting data to or retrieving data from a data source. It is mostly implemented using technologies like HTML/JSP,XML/XSLT, Velocity, Swing, and has been described in the section on Presentation Layers. The controller layer is designed to connect the Model and View layers, and is implemented using technologies like Servelet, JSP, etc.

The MVC has the following advantages of a generic 3-tier architectures.

1. Separation of data from presentation layer
2. Ability of have multiple view of single data
3. It is less painful to change a data layer or business rules
4. Ability to have multiple interface for the same application

MVC differs from the usual 3-tier architectures in topology. In the 3-tier architecture the

communication between the front and back layers is only through the middle layer. However, MVC has a triangular structure wherein the View can send updates to the Controller and itself gets updated directly from the Model part. The Controller can also update the model.

A financial services example
An enterprise application for financial services over the web can allow the visitors to browse through the list of services, create accounts, place orders for the products, and so on. The application may also allow existing customers to change their profile, besides making use of the services offered. Typically, this kind of application has multiple views or screens for the various operations. The business logic represents the user's account, profile, the catalog of services, ordering for services as separate entities, e.g., in the form of Enterprise JavaBeans (EJB).

5.3 Business Logic Layer implementations

Business logic refers to the business rules in a problem domain rather than the view of data or storage. Business Logic Layer provides services to the presentation layer and manipulates the Data layer using appropriate data access logic. It can also make requests to external service

Business Logic layer can be implemented in a variety of ways. It can be depicted as *Business Logic components* residing in the business logic layer which can be directly used by presentation layer, e.g., Enterprise JavaBeans. It can also be encapsulated as a *service* that can be called through a service interface, e.g. Webservices.

Enterprise Java Beans
J2EE architecture is component Architecture for development and deployment of component based distributed business applications written in Java. Enterprise Java Beans (EJBs) are the server-side software components in J2EE architecture which encapsulate business logic.

Web Services
Web services are self-contained, modular applications that can be described, published, located, and invoked over network. They reflect a service-oriented architecture, based on the principle that everything is a service. Web services consist of publishing API for use by other services on network, while hiding implementation details. Communication between services is done using specialised XML based protocols like SOAP. Web Service Architecture is illustrated in Fig. 5.2.

Fig. 5.2 Web Services architecture

Interaction with Presentation Layer

There are various ways by which the code from the business logic layer can interact with the presentation layer. Some of the methods are:

- Interaction between Presentation layer and Business Logic layer using request/response cycle
- Web Browser, Web Server, Java Client (Applets etc), CORBA Clients [Business components]
- Web Client, Web Server, Windows Client etc [Web Services]

Interaction with Data Layer

Business Logic layer interacts with Data Layer for storing, retrieving, manipulating data. The methods to access depend on specific data repository. Standardized languages and methods are available to serve as a common interface to the various data repositories .

The data access technologies include use of SQL, JDBC, Xquery and Xpath. In this book we will mainly illustrate the implementation details using Java and Python.

Design patterns in Server Side (MVC)

A design pattern is a general repeatable solution to a commonly- occurring problem in software design. A design pattern is a template for solving a problem that can be used in many different problems or solutions. Design patterns gained popularity in computer science after the book *Design Patterns: Elements of Reusable Object-Oriented Software* was published in 1994 (Gamma et al).

Design patterns can speed up the development process by providing tested, proven development paradigms. They provide general solutions, documented in a format that doesn't require specifics tied to a particular problem. Design patterns provides developers with well-known, well-understood names for software interactions. Common design patterns can be improved over time, making them more robust than ad-hoc designs.

The design patterns are classified into : Fundamental patterns, Creational patterns, Structural patterns, Behavioral patterns, Concurrency patterns, and Architectural patterns

5.4 Technologies and Framework used in MVC

MVC uses the following technologies :

- Servlet /JSP, XML/XSLT, Velocity, Swing
- JavaBean, Java Data Objects, Data Access Objects

The frameworks used in MVC include the Struts, Turbo Gears and JSF.

Servlets
A Servlet is a java program that runs within the web server. It acts as controller in MVC

architectural pattern. The life cycle of a servlet is given is Fig. 5.3 and has the following components:

- Servlet class is loaded by the container during the start-up
- Container calls the init() method which initializes the servlet. It must be called before the servlet can service any request and is called only once in the entire life of the servlet.
- Client requests are services after initialization, through separate thread for each request. The service()method called for each request, determines the kind of HTTP request (Get, Post, etc.), and accordingly calls the mehtods doGet(), dPost(), etc. The implementation for these methods are provided by the developer. The service() method should never be overloaded.
- The servlets are taken out of service in the end by a container call to the destroy() method.

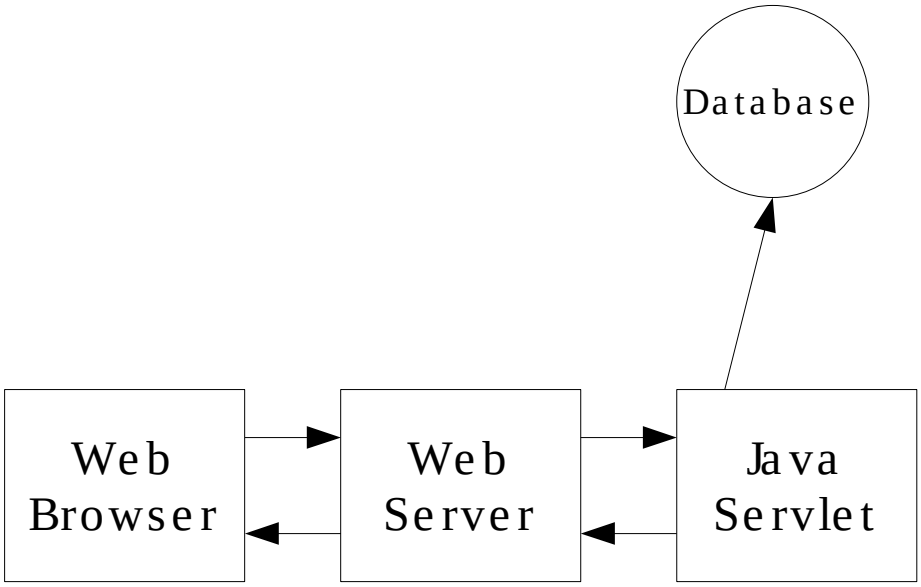

Fig 5.3 Life cycle of a servlet

Java Server Pages

Java Server Pages are web pages with Java code embedded inside them. They are compiled into Java Servlets by JSP compilers. They can be used alone or in conjunction with servlets. JSPs represent (yet) another method for creating server side applications. It also can be used for presentation layer along with HTML.(as view layer)

Servlets v. JSP

The servlets code looks like a regular Java program. On the other hand,

JSP - embed Java commands directly within HTML

Let's examine a Servlet program next to a JSP program… Each of these prints, "Hello, World!"

Servlet version

```
import java.io.;          // A Java Servlet :
import javax.servlet.;    // Looks like a regular
import javax.servlet.http.;// Java program
public class HelloWorld extends HttpServlet {
public void doGet(HttpServletRequest req, HttpSe
     throws ServletException, IOException {
     res.setContentType("text/html");
     PrintWriter out = res.getWriter();
     out.println("<HTML>");
     out.println("<HEAD><TITLE>Hello World</TITLE>
     out.println("<BODY>");
     out.println("<BIG>Hello World</BIG>");
     out.println("</BODY></HTML>");
   }
}
```

JSP version

```
<html>
<head>
  <title>Hello, World JSP Example</title
</head>
  <body>
    <h2> Hello, World!
    The current time in milliseconds is
    <%= System.currentTimeMillis() %>
    </h2>
  </body>
</html>
```

Java Beans

A Java Bean is a reusable software component that can be manipulated visually using a builder tool. Some of the features that distinguish a Java Bean from Java objects are:

- Support for introspection
- Support for customization

- Support for events
- Support for properties
- Support for persistence

Struts

Struts is a web application framework that streamlines the building of the web application based on the MVC design principles. It allows us to fully decouple the business logic, control logic, and presentation code of the applications more reusable and maintainable. Struts also provides libraries and utilities for making MVC development faster and easier.

In Java there are two ways to develop web applications. The normal MVC method is depicted in Fig. 5.4.

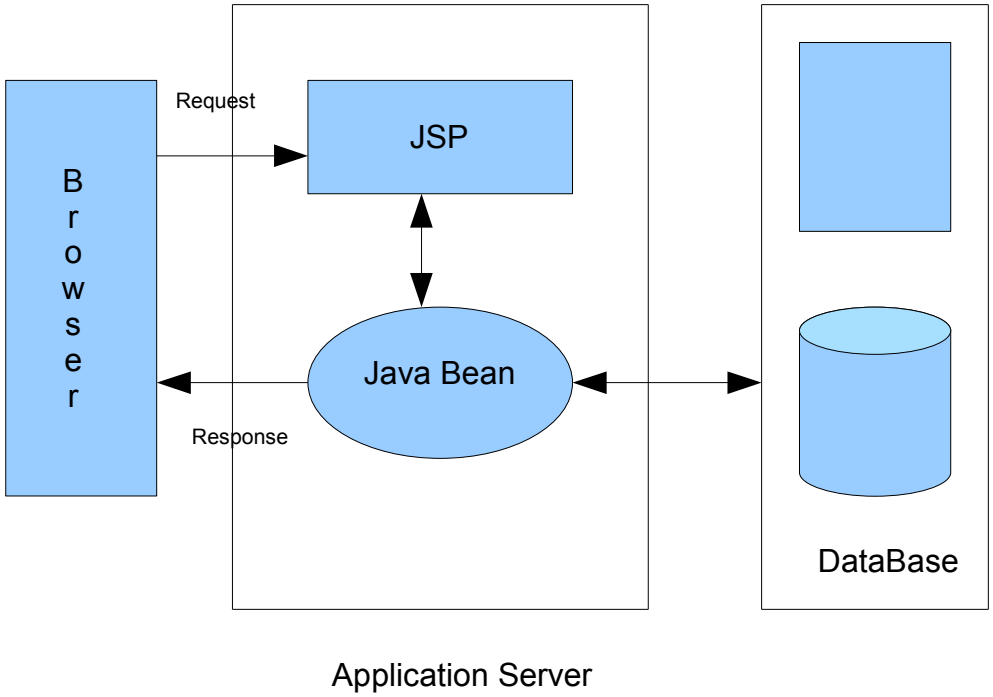

Fig. 5.4 Model 1 Architecture for web applications using Java

The Model 2 architecture, given in Fig. 5.5, is a slight variant of the MVC pattern. Struts encourages the use of Model 2 architecture while developing applications. Struts can be downloaded and installed using instructions from http://struts.apache.org/

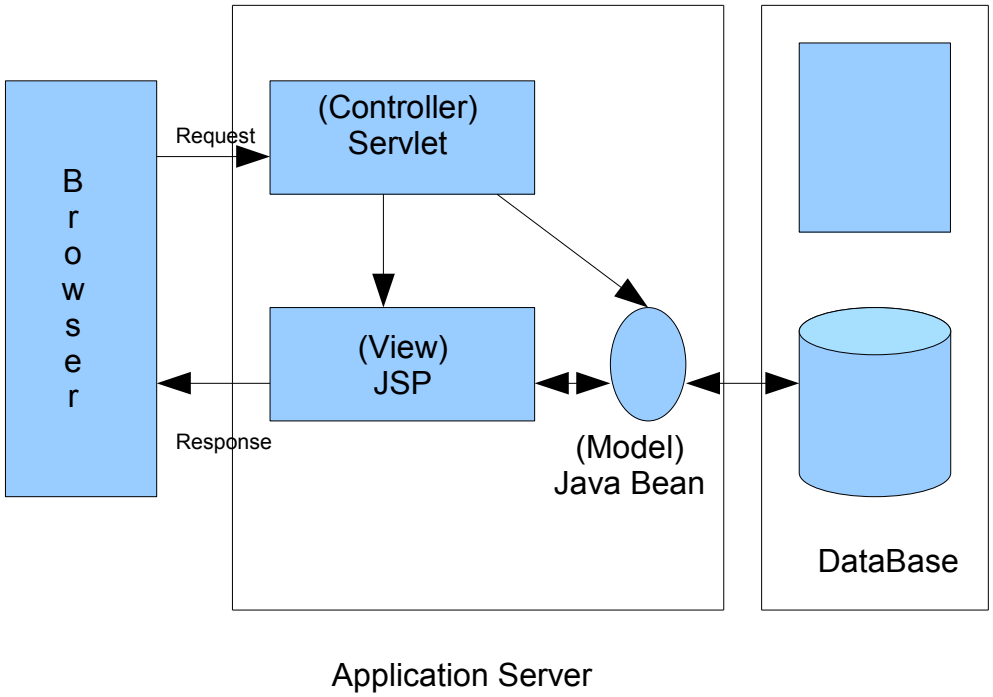

Fig. 5.5 Model 2 Architecture for web applications using Java

Basic component of Struts are :

- Base framework
- JSP tag libraries
- Validator plugin
- Tiles plugin

An example

The sample application will have a basic opening page that links to an employee search page. From the Employee Search page, user can search for employees by name or social security number.After executing the search, the Search page will be redisplayed with a list of employees that match the search criteria.

The flow of execution, as depicted in Fig. 5.6, consist of the following steps:

1. The browser makes a request to the struts application that is processed by ActionServlet (Controller).

2. ActionServlet (Controller) populates the ActionForm(View) object with the HTML form data and invokes its validate() method
3. ActionServlet (Controller)executes the Action object(Controller)
4. Action(Controller) interfaces with the model components and prepares data for view
5. Action(Controller) forwards control to the JSP(View).
6. JSP(view) uses model data to generate a response to the browser.

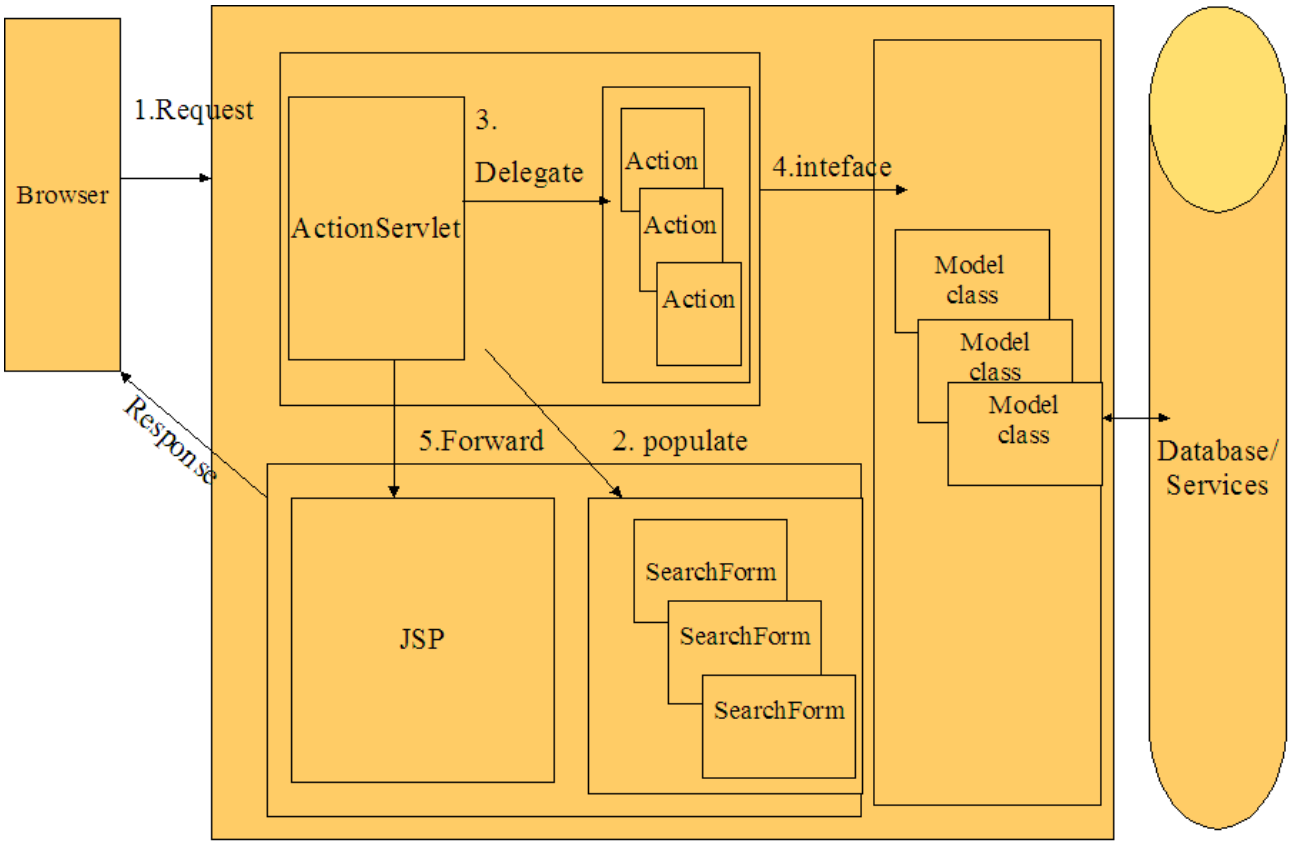

Fig 5.6 Flow of execution of the example of Employee Search page

The model Layer

The model layer is designed to house the business logic and data access code. The MVC architecture dictates that the model Layer should be self contained and function independently from the View and Control layers.That way, the core application code can be used over and over again with multiple user interfaces. The Model Layer break down is given in Fig. 5.7

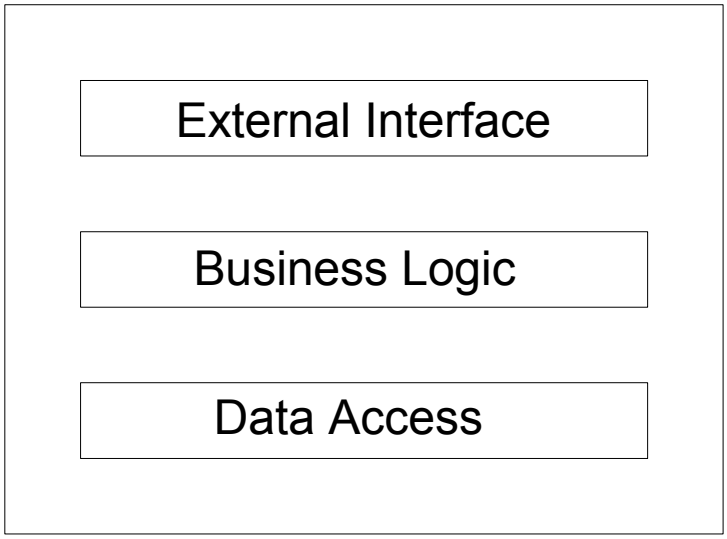

Fig. 5.7 Model Layer break down

The View Layer

The view provides an interface to application. The View layer conduit for getting data in and out of the application. It does not contain business logic. Keeping the Model and View layers separate from one another allows an application's interface to change independent of the Model layer and vice versa.

Struts and the view Layer

There are several forms that the view layer in a struts application can take. It can be HTML/JSP or it can be XML/XSLT, Velocity, Swing or whatever technology the application requires. Since HTML/JSP is the typical View Technology used for the java-based web applications, Struts provides most functionality and features in this way.

Struts HTML/JSP View Layer

JSP pages

- contains the static HTML and JSP library tags that generate dynamic HTML.

Form Beans

- provide the conduit for transferring data between view and controller

JSP tag libraries

- Glues form beans and resource bundles

Resource bundles

- Provide a means of content management

Form Beans

Struts takes the incoming form data and uses it to populate the form's corresponding Form Bean . The Controller layer then uses the Form beans to access data that must be sent to the Model Layer and there is also flip side. Form beans are simple data containers. The form bean life cycle is given below in Fig. 5.8.

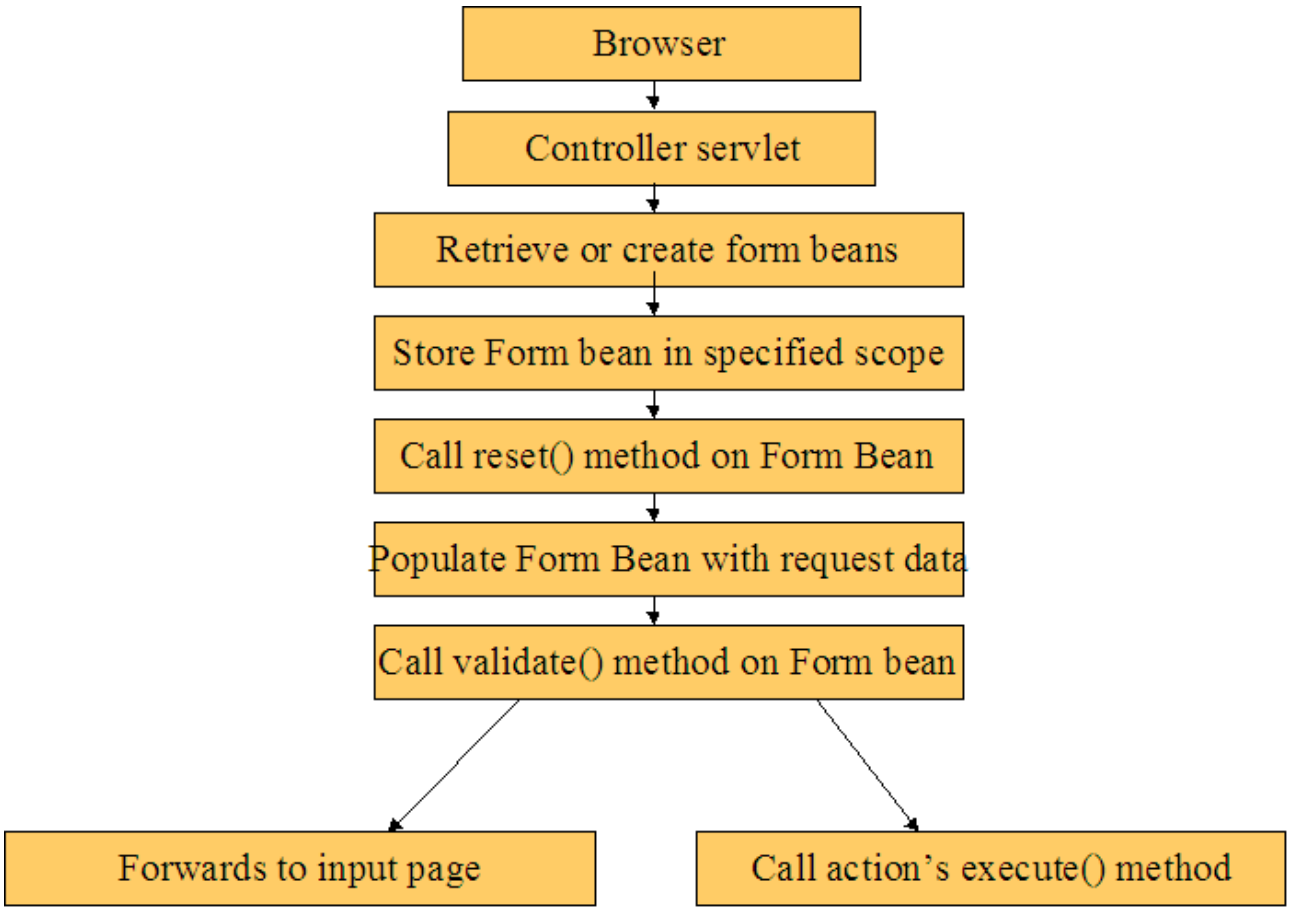

Fig. 5.8 Form bean life cycle

DynaActionForms

Dynamic Form Beans are an extension of Form Beans that allows the specifying of their properties inside the Struts configuration files. Using this allows the properties to be specified in a Struts

configuration file. To change a property, simply update the configuration file. No code has to be recompiled. This eliminates java coding to support HTML forms.

A Code Sample of DynaAction Form is given below:

```
<!– Form beans configuration->
<form-beans>
        <form-bean name="employeeForm"
                type="org.struts.action.DynaActionForm">
        <form-property name="firstName"
                type="java.lang.String" />
        <form-property name="lastName"
                type="java.lang.String" />
        <form-property name="DepartmentName"
                type="java.lang.String" />

</form-beans>
```

JSP Tag libraries

JSP Tag Libraries aid the development of JSPs.

- **HTML**:generate HTML forms with struts api
- **Bean**: intracrt with bean object
- **Logic**:conditional logic in JSPs
- **Nested**: to allow arbitrary level of nesting of the HTML,Bean and Logic Beans

To use a tag library the tag library descriptors are added to the web-xml configuration file as follows:

```
<!-- Struts Tag Library Descriptors -->
<taglib>
  <taglib-uri>/WEB-INF/tlds/struts-html.tld</taglib-uri>
  <taglib-location>/WEB-INF/tlds/struts-html.tld</taglib-location>
</taglib>
```

The following snippets illustrates how your JSPs declare their use of the HTML Tag Library with the JSP taglib directive

```
<%@ taglib uri="/WEB-INF/tlds/struts-html.tld" prefix="html" %>
<html:form cation=/Search>
```

Resource Bundles

Resource bundles allow Java applications to be easily internationalized by having application content placed into bundles. The content can be changed without having to recompile application. Using of resource bundles reduces duplication. They are most commonly used to populate JSP

pages and in customized error messages.

How to use

Create Application resources file as xyz.properties add in classpath. Snippets of this file are given below:

```
page.title=Employee Search
link.employeeSearch= Search for employee
link.addEmployee=Add a New Employee
```

In struts-config.xml

```
<!-- Message Resources Configuration -->
<message-resources    parameter=xyz"/>
```

In jsp page we can access it as

```
<bean:message key=page.title"/>
```

The Controller Layer

The ActionServlet class is the main controller class that receives all incoming HTTP requests for the application. Like other servlet this is configured in the web.xml. The controller life cycle starting with browser through Action Servlet, Request Processor, and action related model and view, is given in Fig. 5.9 below.

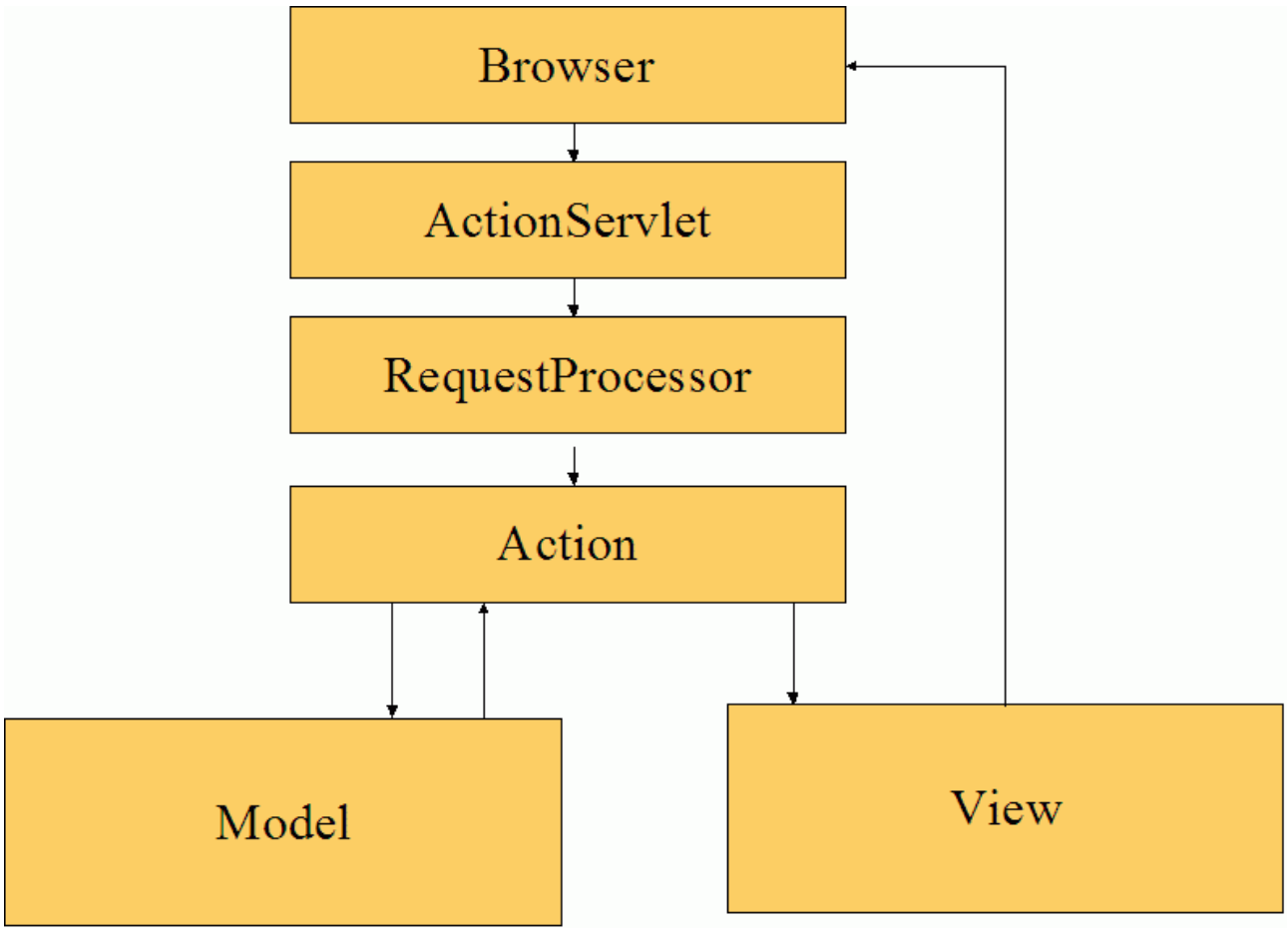

Fig. 5.9 Controller Life Cycle

The RequestProcessor class

Struts uses this class to perform the processing for all requests received by the ActionServlet. To use a custom request processor, one should configure Struts to use it in the Struts configuration file.

The action class

The action class is where struts framework ends and application code starts. There are some Struts Built in Actions .

Validator

Struts captures the form data and use it to populate the ActionForm. The validate() method of the ActionForm then called. If the validation succeeds, processing continues. If an error occurs the HTML form is again displayed. The Validator allow the programmer to focus on only validation code. Capturing data and redisplay incomplete or invalid data is handled by the framework itself.

Shortcoming in validation methods

- Duplication of code for the same type of data in different code
- Any change in validation logic,tends code change in many places and also recompilation

Validator framework

It is a third party add-on to struts, which helps the user to perform declarative validation without java code. The Validate() method vanishes from the ActionForm. Users use a XML configuration file to declare the validations that should be applied to each form bean. If needed, users can plug custom validation into validator. The validator framework supports server side and client side validation where form bean only support server side validation.

Using validator, enabling the Validator plugin is done as follows:

In struts config file

```
<!-- Validator Configuration -->
<plug-in className="org.apache.struts.validator.ValidatorPlugIn">
<set-property property="pathnames" value="/WEB-INF/validator-rules.xml, /WEB-
INF/validation.xml"/>
</plug-in>
```

In validator-rules.xml

```
<validator name="required"
  classname="org.apache.struts.validator.FieldChecks"
  method="validateRequired"
  methodParams="java.lang.Object,
  org.apache.commons.validator.ValidatorAction,
  org.apache.commons.validator.Field,
  org.apache.struts.action.ActionMessages,
  javax.servlet.http.HttpServletRequest"
msg="errors.required"/>
```

Inside the global tag, extend the SearchForm from ValidatorForm. Remove validate()/reset() method.

Configure form bean

```
<!-- Form Beans Configuration -->
<form-beans>
  <form-bean name="searchForm" type="in.ernet.ncst.akshay.example2.SearchForm"/>
</form-beans>
```

In validation.xml file

```
<form-validation>
<formset>
  <form name="searchForm">
    <field property="ssNum" depends="mask">
      <arg0 key="label.search.ssNum"/>
      <var>
        <var-name>mask</var-name>
        <var-value>^\d{3}-\d{2}-\d{4}$</var-value>
      </var>
    </field>
  </form>
</formset>
</form-validation>
```

If we are using the action.DynaActionForm, extend it by validator.DynaValidatorForm

A scenario

Validation definition is tied to Form Bean. It can not validate a subset of the fields of the form. The solution is to extend validator.ValidatorActionForm or DynavalidatorActionForm for the form. To do this, in validation.xml, we map a set of validations to an action path instead of a Form Bean name. This is necessary because if we have two actions defined – e.g., Create Address and Edit Address which use the same Form Bean -- each will have a unique action path.

Tiles

Tiles can be thought of as a technology similar to CSS. With Tiles, one can define a master layout JSP that specifies each of the includes that fill in the layout and then one could define which content should fill in the layout in an external configuration file. The same layout can be used over and over by simply specifying different filler content for the layout in the configuration file.

Using Tiles

Enable the Tiles plug-in(in struts-config file)

```
<!-- Tiles Configuration -->
<plug-in className="org.apache.struts.tiles.TilesPlugin">
  <set-property property="definitions-config" value="/WEB-INF/tiles-defs.xml"/>
</plug-in>
```

Create Tiles Definitions is done as follows :

XML configuration File

```
<tiles-definitions>
```

```
<!-- Main Layout -->
<definition name="main.layout" path="/mainLayout.jsp">
  <put name="title"  value=""/>
  <put name="header" value="/header.jsp"/>
  <put name="body"   value=""/>
  <put name="footer" value="/footer.jsp" />
</definition>

<!-- Search Page -->
<definition name="search.page" extends="main.layout">
  <put name="title"  value="Search Page"/>
  <put name="body"   value="/search.jsp"/>
  </definition>
</tiles-definitions>
```

In the jsp page:

```
<tiles:insert definition=search.page"/>
```

JSP

```
<@taglib uri=/WEB-INF/struts-tiles.tld prefix=tiles%>
<tiles:definition id=search.page template=/mainLayout,jsp>
  <tiles:put name=header value=/header.jsp/>
  <tiles:put name=body value=/search.jsp/>
  <tiles:put name=footer value=/footer.jsp/>
</tiles:definition>
```

To use this jsp based definitions,

```
<tiles insert beanName=search.page flush=true/>
```

Creating Layout JSPs and Using the Tiles Tag Library

Header.jsp

```
<font size="+1">ABC, Inc. Human Resources Portal</font><br>
<hr width="100%" noshade="true">
```

Declarative Exception Handling

Declarative Exception Handling is a mechanism that lets you specify how an exception should be handled, but without having to include the handler within your application's java code. Instead, the exception handler is declared Struts XML based configuration file, struts-config.xml

Create an application exception class

```
package com.jamesholmes.minihr;
public class NoResultsFoundException extends Exception
{
}
```

Update the Action class to the application exception

```
import org.apache.struts.action.ActionMapping;
public final class SearchAction extends Action{
  public ActionForward execute()throws Exception{
  ..
  // Throw an application exception if results were not found.
  if (results.size() < 1) {
    throw new NoResultsFoundException();
  }
  }
}
```

Configuration of Struts's Exception Handler

```
<!-- Action Mappings Configuration -->
  <action-mappings>
    <action path="/search"
      type="com.jamesholmes.minihr.SearchAction"
      name="searchForm"
      scope="request"
      validate="true"
      input="/search.jsp">
    <exception
      type="com.jamesholmes.minihr.NoResultsFoundException"
      key="error.NoResultsFoundException"
      path="/exception.jsp"/>
    </action>
  </action-mappings>




  <global-exceptions>
    <exception
    type="com.jamesholmes.minihr.NoResultsFoundException"
    key="error.NoResultsFoundException"
    path="/exception.jsp"/>
  </global-exceptions>
```

Creating an exception-Handler JSP

Exception.jsp

```
<%@ taglib uri="/WEB-INF/tlds/struts-html.tld" prefix="html" %>
<html>
<head>
```

```
  <title>ABC, Inc. Human Resources Portal</title>
</head>
<body>
  <font size="+1">ABC, Inc. Human Resources Portal</font><br>
  <hr width="100%" noshade="true">
  <html:errors/>
</body>
</html>
```

Add an Exception Error Message to the ApplicationResources.properties File

```
# Exception Error Resources
error.NoResultsFoundException=<li>No Search Results Found</li>
```

Struts Modules

Using Struts modules allows partition of Struts application into discreet functional areas, almost like having wholly separate applications. Each module has its own Struts application file, actions, forwards, and JSPs, which allows it to function independently of other modules. Using modules in a Struts application involves the following three steps:

1. Create a Struts configuration file for each module.

2. Configure the web.xml deployment descriptor for modules.

```
<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
  <init-param>
  <param-name>config</param-name>
  <param-value>/WEB-INF/struts-config.xml</param-value>
  </init-param>
  <init-param>
  <param-name>config/employee</param-name>
  <param-value>/WEB-INF/struts-config-employee.xml</param-value>
  </init-param>
  <init-param>
  <param-name>config/reports</param-name>
  <param-value>/WEB-INF/struts-config-reports.xml</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
```

3. Configure links to access module-specific JSPs.

```
<action path=/switchMod type=org.apache.struts.actions.SwitchAction />
...
<html:link action=/switchMod?prefix=/ModuleA&page=/moduleA/main.jsp>moduleA main
JSP</html:link>
```

Using validator and Tiles with Modules

- Add the Validator plugin to each modules-specific ValidatorPlugin to each modules Struts configuration file.
- Add the Tiles plugin to each modules-specific TilesPlugin to each modules Struts configuration file.

Internationalizing Struts Applications

Java's built-in internationalization support is centered on three main classes. Table 5.1 lists each class and its description

Class	Description
Java.util.Locale	Encapsulates the languages,country, and varient for a specific locale
Java.util.resourcebundle	Encapsulates locale specific resources
Java.text.MessageFormat	Provides methods for creating locale-specific formatted messages

Table 5.1 Java classes related to built-in intenationalization support

Like java, Struts's internationalization support is centered on the Locale class and resource bundles.

The Struts org.apache.struts.util.Message Resources and org.apache.struts.util. Property Message Resources, provide functionality parallel to the ResourceBundle and PropertyResourceBundle classes built into java.

Because Java's ResourceBundle classes are not serializable, Struts has created its own classes that duplicate their functionality and are serializable.

Steps for the internationalizing application

- Add entries for all application text to the ApplicationResources.properties file.
- Create spanish version of the ApplicationResources.properties file.
- Update JSPs to retrieve all application text from the ApplicationResources.properties file.

Constraints for internationalization

Please note that the i18n support in a framework like Struts is limited to the presentation of internationalized text and images to the user. Support for Locale specific input methods (used with languages such as Japanese, Chinese, and Korean) is left up to the client device, which is usually a

web browser.

There are two final steps to ensure that your application can fully support Unicode.

1. The first is to make sure your database is configured for Unicode. This is usually the default, but you should verify this is the case.
2. Second, if you are using the JRE rather than the SDK, use the I18N version and not the U.S. version.

DBCP package

The goal of this sub-project is to provide a reusable JDBC Datasource that can help us manage to all of our database connections. It can be used either in a stand-alone fashion or as a Datasource integrated into Struts.

To initialize the DBCP, we must add a new <data-source> element to the struts-config.xml file.

```
<struts-config>
  <data-sources>
  <data-source type =org.apache.commons.dbcp.BasicDataSource>
  <set-property property=driverClassName value=..>
  <set-property property=url value=..>
  <set-property property=username value=..>
  <set-property property=password value=..>
  </data-source>
  </datasources>
</struts-config>
```

The datasource retrieving is a simple process. We pass the getDataSource() method a reference to the request and it pulls the DBCP DataSource out of the ServletContext. Once we have a reference to the DBCP DataSource, we can get a Connection object from it and continue with normal JDBC processing. While the returned Connection looks and acts just like any other connection object,it actually a wrapper object around a java.sql.Connection object. The purpose of this wrapped Connection is to allow the Datasource to manage the Connections stored in its pools. In normal definition of the Connection class, close() method would close the connection to the database and thus render it useless for the later processes. Because this Connection object is an instance of a wrapper object, however, the Connection is returned to the pool for later use instead of being closed.

Extending the Struts Framework

Framework extension points, also referred to as hooks, allow the user to extend the framework in specific places to adapt it to meet the application's requirements. Where and how a framework provides these hooks is very important. If they are provided incorrectly or in wrong locations, it becomes very hard for an application to adapt the framework, which makes framework less useful.

General extension points

The following are some of the points where the framework can be extended.

- Using the Plugin Mechanism
  Shown in validator example and tiles
- Extending the ActionServlet class
- Extending the RequestProcessor class
- Extending the Base action class
- Extending Struts Custom Tags
- Extending Model Components

JDBC

JDBC defines an API that allows an application to access virtually any tabular data. It is a call-level SQL interface for Java SQL conformance which was influenced by X/OPEN SQL Call Level interface. Note: JDBC is a trademarked name, however, it is often thought of as standing for Java Database Connectivity.

Architecture

The JDBC API contains two major sets of interfaces:

1. JDBC API for application writers : The public API to be used by programmers who want to use JDBC to connect their Java programs to databases.

2. Lower-level JDBC driver API for drivers writers : This is the internal API to be followed by programmers who will be writing JDBC drivers for databases. Drivers are client-side adaptors (they are installed in the client machine, not in the server) that convert requests from Java programs to a protocol that the DBMS can understand.

JDBC Drivers types

There are Four categories of JDBC drivers:

Type 1: JDBC-ODBC bridge plus ODBC driver

The JDBC type 1 driver, also known as the JDBC-ODBC bridge is a database driver implementation that employs the ODBC driver to connect to the database. The driver converts JDBC method calls into ODBC function calls. The bridge is usually used when there is no pure-Java driver available for a particular database. This driver comes with the Java 2 SDK, Standard Edition.

The driver is platform-dependent as it makes use of ODBC which in turn depends on native libraries of the operating system. Also, using this driver necessitates that other dependencies such as

ODBC must be installed on the computer having the driver and the database which is being connected to must support an ODBC driver. Hence the use of this driver is discouraged if the alternative of a pure-Java driver is available.

Type 1 is the simplest of all but platform specific, i.e., only to Microsoft platform.

Type 2: Native API partly-Java driver

The JDBC type 2 driver, also known as the Native-API driver is a database driver implementation that uses the client-side libraries of the database. The driver converts JDBC method calls into native calls of the database API.

The type 2 driver is not written entirely in Java as it interfaces with non-Java code that makes the final database calls. The driver is compiled for use with the particular operating system. For platform interoperability, the Type 4 driver, being a full-Java implementation, is preferred over this driver.

However the type 2 driver provides more functionality and performance that the type 1 driver as it does not have the overhead of the additional ODBC function calls.

Type 3: JDBC Net pure Java driver

The JDBC type 3 driver, also known as the network-protocol driver is a database driver implementation which makes use of a middle-tier between the calling program and the database. The middle-tier (application server) converts JDBC calls directly or indirectly into the vendor-specific database protocol.

This differs from the type 4 driver in that the protocol conversion logic resides not at the client, but in the middle-tier. However, like type 4 drivers, the type 3 driver is written entirely in Java.

The same driver can be used for multiple databases. It depends on the number of databases the middleware has been configured to support. The type 3 driver is platform-independent as the platform-related differences are taken care by the middleware. Also, making use of the middleware provides additional advantages of security and firewall access.

Type 4: Native protocol pure Java driver

The JDBC type 4 driver, also known as the native-protocol driver is a database driver implementation that converts JDBC calls directly into the vendor-specific database protocol.

The type 4 driver is written completely in Java and is hence platform independent. It is installed inside the Java Virtual Machine of the client. It provides better performance over the type 1 and 2 drivers as it does not have the overhead of conversion of calls into ODBC or database API calls. Unlike the type 1 and 2 drivers, it does not need associated software to work.

As the database protocol is vendor-specific, separate drivers, usually vendor-supplied, need to be used to connect to the database.

The JDBC API package lies underjava.sql.*

Using JDBC: Basic Steps

1. Load a JDBC driver
2. Define the connection URL
3. Establish the connection
4. Create a statement object
5. Execute a query or update
6. Process the results
7. Close the connection

Load the driver

The JDBC driver is loaded by specifying a driver, which would be talking to the database. The driver class name is provided by the vendor. The driver can be called in two ways:

1. By calling the method Class.forName
2. By using the jdbc.drivers system property

Loading the driver, creates instance of the driver and registers it with the DriverManager.

Define the Connection URL

The connection to the database is specified using the following standard syntax irrespective of the database.

jdbc:<subprotocol>:<subname>

The standard syntax, provides a way of identifying the data source. The exact name is defined by the drivers providers, confirming to the recommendation. The URL information will be found in the driver documentation.

<subprotocol>: name of the driver or the name of a database connectivity mechanism

<subname>: varies, depending on the subprotocol, and can have any internal syntax, chosen by the driver provider.

Establish the connection

Establishing the connection is done by calling one of the DriverManager's getConnection() methods

getConnection(String url)
getConnection(String url, String user, String password)
getConnection(String url, java.util.Properties info)

The DriverManager finds a driver from its pool of loaded drivers, and that driver establishes a connection to the specified datasource and returns a Connection object.

Create a statement object

SQL queries to database are through statements which are encapsulated in a statement object.

There are three types of statements :-

- Statement -- Useful for executing static SQL statements, and obtaining results produced by it
  Statement statement = connection.createStatement();

- PreparedStatement -- Represents a pre-complied statement, which can be executed multiple times, with varying parameters. The IN values are set using various setXxx() methods on the PreparedStatement object

  String sql = SELECT * FROM Students WHERE S_ID = ?;
  PreparedStatement statement = connection.prepareStatement(sql);
  statement.setString(1, new String(FPGDST));

- CallableStatement -- Used to execute SQL stored procedures

Execute a query or update

Once we have built the statement, we need to execute it. This is done using methods on Statement Object to execute an SQL statement.

executeQuery()
Used to perform SELECT query on the database, and retrieve a set of records in the form of a ResultSet

executeUpdate()
Used to execute an SQL DDL statements, like INSERT, UPDATE, DELETE etc

Process the results

The output of the SQL statement are in the form of ResultSets. A ResultSet is a row of data returned by a database query. The columns are retrieved using the getXxx() methods.

Close the connection

When through, always call close() to close the connection to the database.

```
try {
    // ..
    // closing the ResultSet
    result.close();
    // closing the Statement
    statement.close();
    // closing the Connection
    connection.close();
```

```
    // ..
    } catch (SQLException sqle) {
    // handle the SQL exception here
}
```

5.5 Jasper Reports

A reporting tool is a software which allows end users to prepare professional business reports, make reports based on data in a database and allow scheduled runs for automatic running and distribution .

Jasper Reports is an open source Java library for report generation. It can be used for creating page oriented, ready to print documents. It can also be used to deliver content on screen or to the printer. It also has the facility to output reports in various formats like PDF, HTML, CSV and XML.

Written entirely in Java and licensed under the GNU Lesser Public License, Jasper Reports are used in a variety of Java enabled applications, from full fledged J2EE application to small web applications.

From a user perspective, Jasper Reports can be thought of as a library which provides a set of packages, which provide functionality to develop reports. It is available from http://jasperreports.sourceforge.net . It is enough to get the jar files from the website, and include them in the classpath to start using them. There are a number of sample reports in the download to get started.

Creating a Report

Preparing Reports in Jasper Report

1. Design Report in xml sticking to the predefined DTD
2. Save the report with .jrxml extension
3. Compile the report design using java code. This is a one time job
4. Get the compiled reports object
5. Fill it with data from a data source
6. Catch the returned, ready to print object
7. Display, Print
8. Export in some format and store

Features of Jasper Reports

Report Designs

Report Designs are templates used by Jasper Reports, to dynamically fill in with data from a variey of data sources. They are written in XML with Jasper Report's DTD.

```
<!DOCTYPE jasperReport
  PUBLIC "-//JasperReports//DTD Report Design//EN"
```

"http://jasperreports.sourceforge.net/dtds/jasperreport.dtd">

The report designs have an extension of .jrxml

Some of the basic tags are :

- <jasperReport> - the root element
- <title> - its contents are printed only once at the beginning of the report
- <pageHeader> - its contents are printed at the beginning of every page in the report.
- <detail> - contains the body of the report.
- <pageFooter> - its contents are printed at the bottom of every page in the report.

Parameter

Parameters are Object References which are passed during the report filling operation. Parameters are used to pass data that is not available with the data source and are used in the query string of the report to act as dynamic filter for the data.

Defining a parameter

```
<parameter
    name=name
    class=class
/>
```

Referring a parameter is done by :

```
$P{name}
```

Expressions

Jasper Report expressions are basically Java expressions. They can reference report fields and report variables. They are used for report variable declaration, text field contents specification and data grouping.

Variables

Variables are built over expressions

```
<variable
name=name        class=class"
resetType=reset type
calculation=built in calculation">
<variableExpression> Expression </variableExpression>
</variable>
```

Variables simplify report designs. They are declared only once and can be used to perform calculations.

Scriptlets

Scriptlets are sequence of Java codes which are called when a report event occurs. They can be created by extending the class

net.sf.jasperreports.engine.JRAbstractScriptlet

or

net.sf.jasperreports.engine.JRDefaultScriptlet

Some of the common event which can be handled by scriptlets are:

- Report init
- Page init
- After page init
- After report init

Groups

Groups are used to organize data on report. The user has to define a condition on which grouping is done. When a group ends group header and group footer are automatically inserted. A group can also have other subgroups in it. The hierarchy of groups is defined by the order amongst the declarations in .jrxml file.

Charts

Jasper Reports has a predefined chart component based on JfreeChart. Configuring and using a chart involves the following components

- Overall chart component
- Chart dataset
- Chart plot

Many different chart types are supported like:

- Pie
- Pie 3D
- Bar
- Bar 3D
- etc.

Example

This simple example is based on the fonts demo application included with the Jasper Reports distribution.
The jrxml file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jasperReport PUBLIC "-//JasperReports//DTD Report Design//EN"
"http://jasperreports.sourceforge.net/dtds/jasperreport.dtd">

<jasperReport name="FontsReport" language="java" leftMargin="20" rightMargin="20"
topMargin="30" bottomMargin="30">
        <title>
                <band >
                        <staticText>
                                <reportElement x="0" y="50" width="150" height="40"/>
```

```
                                <textElement/>
                                <text><![CDATA[The quick brown fox jumps over the lazy
dog.]]></text>
                        </staticText>
                        <staticText>
                                <reportElement x="160" y="50" width="390" height="40"/>
                                <textElement/>
                                <text><![CDATA[<staticText><reportElement x="0" y="50"
width="150" height="40"/><text>The quick brown fox jumps over the lazy
dog.</text></staticText>]]></text>
                        </staticText>
                </band>
        </title>
</jasperReport>
```

This report has to be compiled only once to be reused. It can be done by customising the ANT build scripts provided with the jasper reports distribution. Compiling a report mainly consists of sending the name of the jrxml to the method :

net.sf.jasperreports.engine.**JasperCompileManager.compileReport**

Using this report to generate HTML output can be done with the following code fragment:

```
File sourceFile = new File(fileName);

JasperPrint jasperPrint = (JasperPrint)JRLoader.loadObject(sourceFile);
File destFile = new File(sourceFile.getParent(), jasperPrint.getName() + ".html");
JRHtmlExporter exporter = new JRHtmlExporter();

HashMap fontMap = new HashMap();
fontMap.put("sansserif", "Arial, Verdana, Tahoma");
fontMap.put("serif", "Times New Roman");
fontMap.put("monospaced", "Courier, Courier New");

exporter.setParameter(JRExporterParameter.JASPER_PRINT, jasperPrint);
exporter.setParameter(JRExporterParameter.OUTPUT_FILE_NAME, destFile.toString());
exporter.setParameter(JRExporterParameter.FONT_MAP, fontMap);

exporter.exportReport();
```

Similarly export to other formats can be done using the appropriate formatters.

Jasper Reports for other software

Jasper Reports can be used with SugarCRM, an open source customer relationship management solution. It provides full support to SugarCRM, provides a repository of reports that can be run with SugarCRM database and allows easy addition of new reports. Many other software can also use Jasper Reports as their reporting engine.

Tools for Jasper Reports

- iReport Designer
  Visual report builder/designer for Jasper Reports
  Written in java
  Allows users to

- visually edit complex reports with charts, images, etc
- No need to go for XML

5.6 TurboGears

Till now we have seen how to develop MVC based applications in Java. Now we take a look at developing MVC applications in Python. This is to emphasise the point that MVC pattern is not restricted to any specific language. The framework we are going to look into is TurboGears -- http://turbogears.org. Since it is a rapidly developing framework, for the latest download and installation instructions, look at the website.

The TurboGears website describes TurboGears as "A Python web mega framework". It is called so because it is a collection of matured components which are stitched together. All the components were developed before TurboGears came into existence. The components of TurboGears are:

- SQLObject
  Model : Manipulates the backend databases.
- CherryPy
  Controller : Event handler for web based requests.
  Includes an embedded web server.
- Kid
  View : A template based language.
- MochiKit
  A powerful pythonic JavaScript libray.

SQLObject

SQLObject can be used to use databases without writing SQL. It acts as an Object Relational Mapper. Classes are mapped to tables, attributes to columns and instances to rows. SQLObject generates code to access the database and update any changes made. Creating an object inserts a new row in the database. It also provides a python object based query language.

Example:

```
Person.select(Person.q.firstName=Turbo)
object.set(firstName=Gears)
```

The attribute "q" gives access to special objects (column) for constructing query clauses. Creating an object inserts a new row in the database

CherryPy

CherryPy is a Pythonic, object-oriented web development framework. It makes building web applications as close as building any other object-oriented Python program. It includes a light embedded web server. It does not include a template language.

Kid

Kid is a simple Python based XML template language. Kid template files are well-formed XML documents. The templates can embed Python code for generating and controlling dynamic content. Kid can generate many kind of XML documents including XHTML, RSS, Atom, SOAP, etc.

MochiKit

Mochikit is a well documented and well tested suite of JavaScript libraries. Mochikit has been designed such that its libraries are natural to use for both Python and JavaScript programmers. It is mostly used for implementing AJAX features that uses JSON (JavaScript Object Notation) data stream for transferring data. JSON represent data as nested hash tables, lists and simple values. JSON has a significant advantage over XML: the syntax is native JavaScript so it's fast when compared to XML.

Installation

1. Download ez_setup.py script file from the projects main website.

2. Executing the script will download and install all the required packages needed to make a decent web application.

3. More packages can be downloaded and installed as per requirement.

20 minutes Wiki Tutorial

This is a turbogears tutorial based on the 20 minute wiki present in the turbogears website. This tutorial is also available as a screencast video from the website. It is recommended to download and watch the video to understand this tutorial better.

The Quickstart

TurboGears has a command line tool with a few features that will be touched upon in this tutorial. The first is quickstart, which will get us up and running quickly:

```
tg-admin quickstart
```

Running the above command will provide a prompt for the name of the project (this is the name by which the application will be known), and the name of the package (this is the internal name for the project. For this tutorial, we'll be using these values:

```
Enter project name: Wiki 20
Enter package name [wiki20]: wiki20
```

Do you need Identity (usernames/passwords) in this project? [no] no

This creates a few files in a directory tree just below the current directory.

cd wiki20

The file start-wiki20.py is the startup script for the built-in web server. It can be started with

python start-wiki20.py

By default turbogears applications are set up to run at the 8080 port. To view the current application, open http://localhost:8080/. It will currently have the default welcome page with the current time (fig. 5.10).

Fig. 5.10 TurboGears default welcome page

The Default Code

In addition to the startup script, there are a few more interesting files which have been created.

1. wiki20/controllers.py has the code that's generating the welcome page. CherryPy enables the user to write methods and expose them to the web. TurboGears adds the automatic template processing to go from a dictionary of values to HTML for the browser.
2. wiki20/templates/welcome.kid is the template for the welcome screen. It is standard XHTML with some simple namespaced attributes. It can even be opened directly in a browser.

The Model

As we have seen previously in TurboGears, Kid templates are the view, CherryPy classes are the controllers and basically any object *can* be the model. In practice, since we are in a database-driven world, the model objects will be SQLObjects.

TurboGears quickstart gives a model.py module with enough in it to start creating model classes. The new Page class:

```
class Page(SQLObject):
    pagename = UnicodeCol(alternateID=True, length=30)
    data = UnicodeCol()
```

Here the database will be created based on the definition of the objects. SQLObject *does* support creating objects based on the database definition. Since the pagename has alternateID=True, it is guaranteed that the column will be unique which allows searching on pagename. Some databases require the length to be specified on columns which are to be indexed. The length 30 here is arbitrary.

Pointing to a database

TurboGears has a minimum of required configuration. It *does* need to know where the database lives. The quickstart creates a couple of simple config files which can be modified to suit the project requirements. In this instance the database being used in sqlite. It can be installed from the distribution's repositories.

The configuration file for a development environment is the "dev.cfg" file. In the file sqlobject.dburi line should be uncommented and made to point to the database, with the proper connection info. With sqlite, it does not matter if the database file does not exist yet. It will be created soon.

Restart the web server by hitting control-C and running the startup script again:

```
python start-wiki20.py
```

Creating the Database

The schema for the database and its location has already been specified. To create the database the following command is to be used:

```
tg-admin sql create
```

The "tg-admin sql" command is a wrapper around SQLObject's sqlobject-admin command. It looks in the config file to figure out where to find the database.

Displaying a wiki page

The templates are renamed to match their uses.

```
cd wiki20/templates
mv welcome.kid page.kid
cd ../..
```

The body of the template has to be replaced with a reasonable introduction for a wiki page:

```
<div style="float:right; width: 10em">
Viewing <span py:replace="page.pagename">Page Name Goes Here</span>
<br/>
You can return to the <a href="/">FrontPage</a>.
</div>

<div py:replace="XML(data)">Page text goes here.</div>
```

TurboGears greatly reduces the amount of code needed, but it does not eliminate it. To enable it display the new template as the default for the application, the file controllers.py has to be modified.

In the top of the file we add,

```
from wiki20.model import Page
from docutils.core import publish_parts
```

Then, we'll replace the index method with one that:

1. Sets the template to our newly named "page" (line 1)
2. Has a default pagename of "FrontPage" (line 2)
3. Retrieves the page from the database (line 3)
4. Formats the text as HTML (line 4)
5. Returns the data for the template to use (line 5)

Here is the new index method:

```
@expose("wiki20.templates.page")
def index(self, pagename="FrontPage"):
    page = Page.byPagename(pagename)
    content = publish_parts(page.data,
                            writer_name="html")["html_body"]
    return dict(data=content, page=page)
```

The dictionary that is returned at the end provides the data that populates the template. The "." at the beginning of the template name says that this name is relative to the controller's package. In other workds, this is referring to the full name of "wiki20.templates.page".

Now the code to display the wiki frontpage is ready. The page has to be created. The page can be created in the database using CatWalk, the model browser:

tg-admin toolbox

- A new browser window will open with the TurboGears toolbox (Fig. 5.11).
- Click on the CatWalk model browser. You'll see "Page" listed on the left. (Fig. 5.12)
- Click on that, select the "Add Page" tab (Fig. 5.13)
- Create a page with the name "FrontPage" and with whatever text you want for the data. (Fig. 5.14)

That's all there is to it. We just created a new page in the database (Fig. 5.15).

Fig 5.11 TurboGears ToolBox

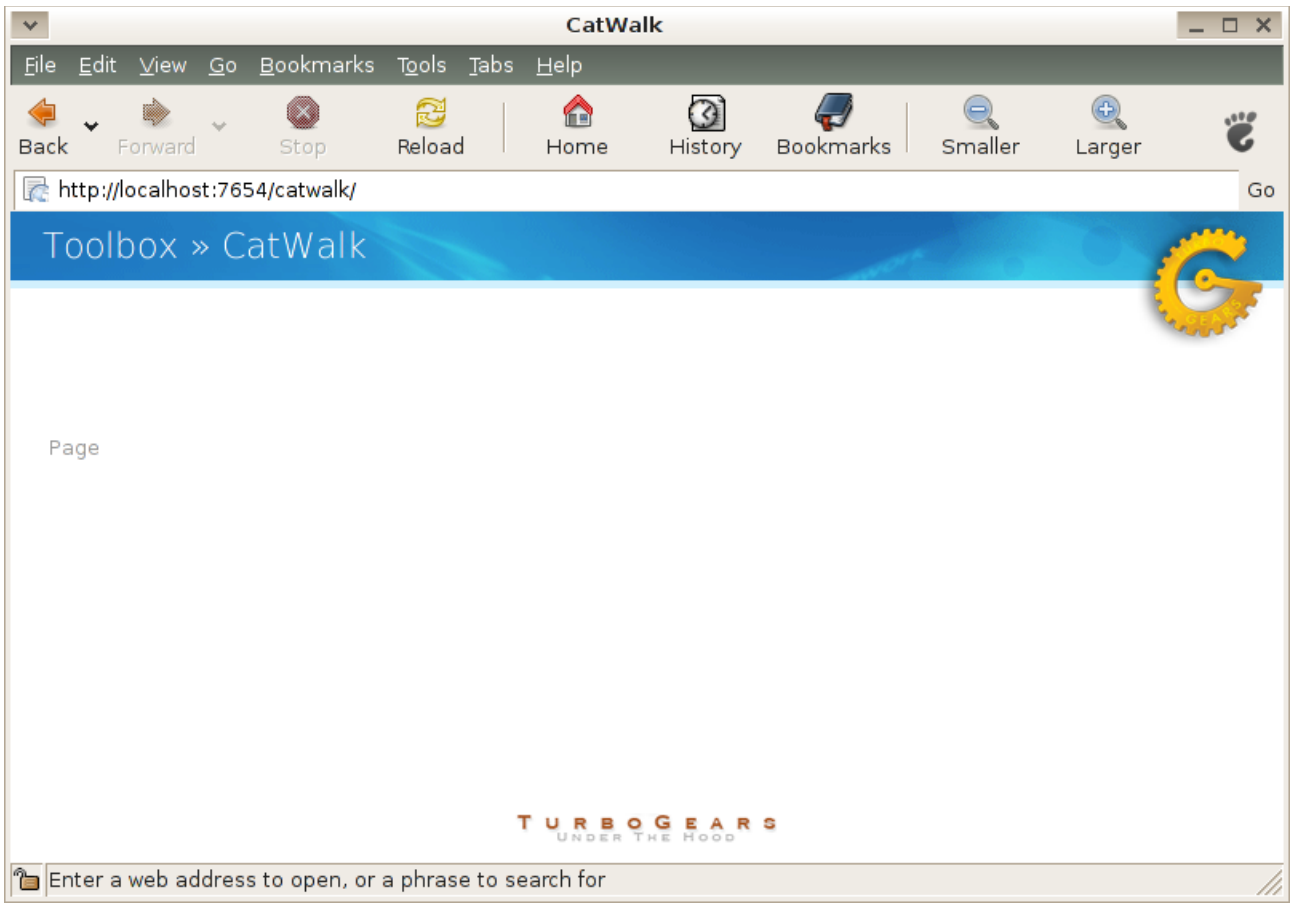

Fig. 5.12 TuboGears Toolbox : Catwalk

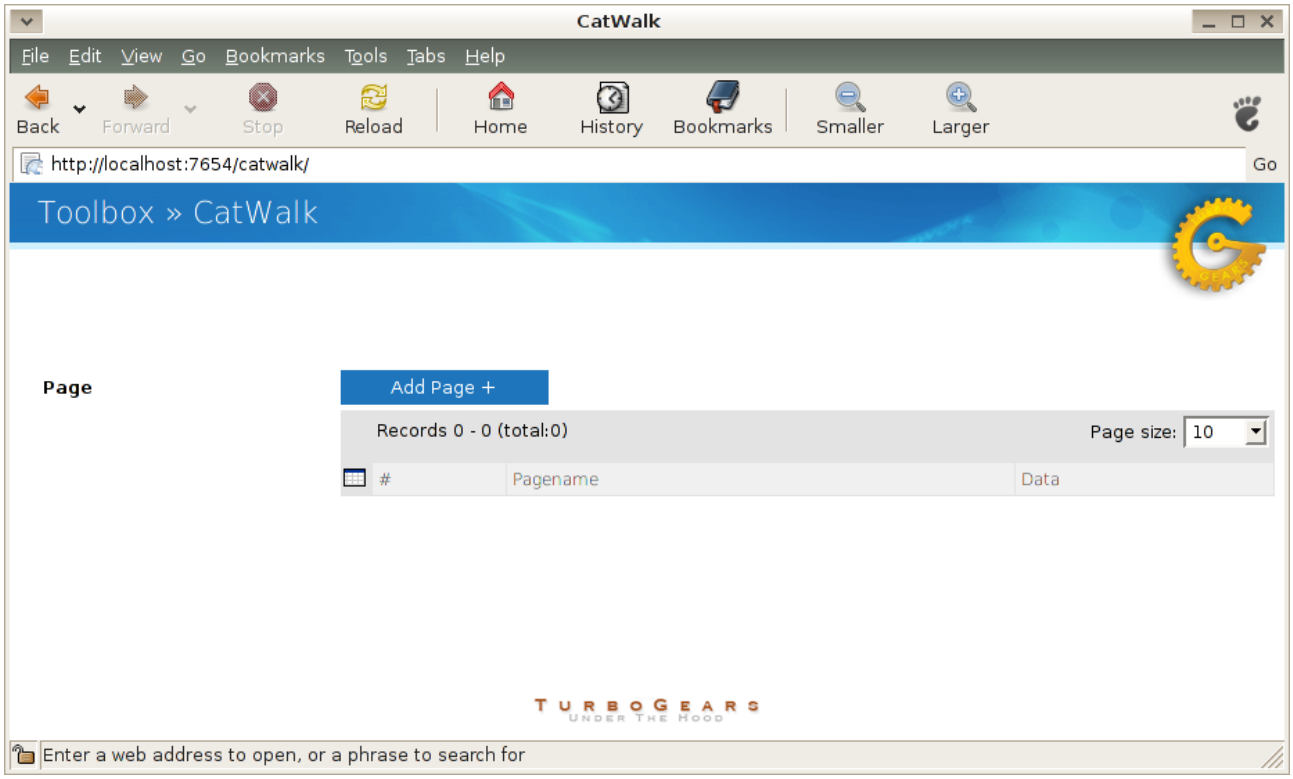

Fig. 5.13 TuboGears Toolbox : Catwalk – opening a page

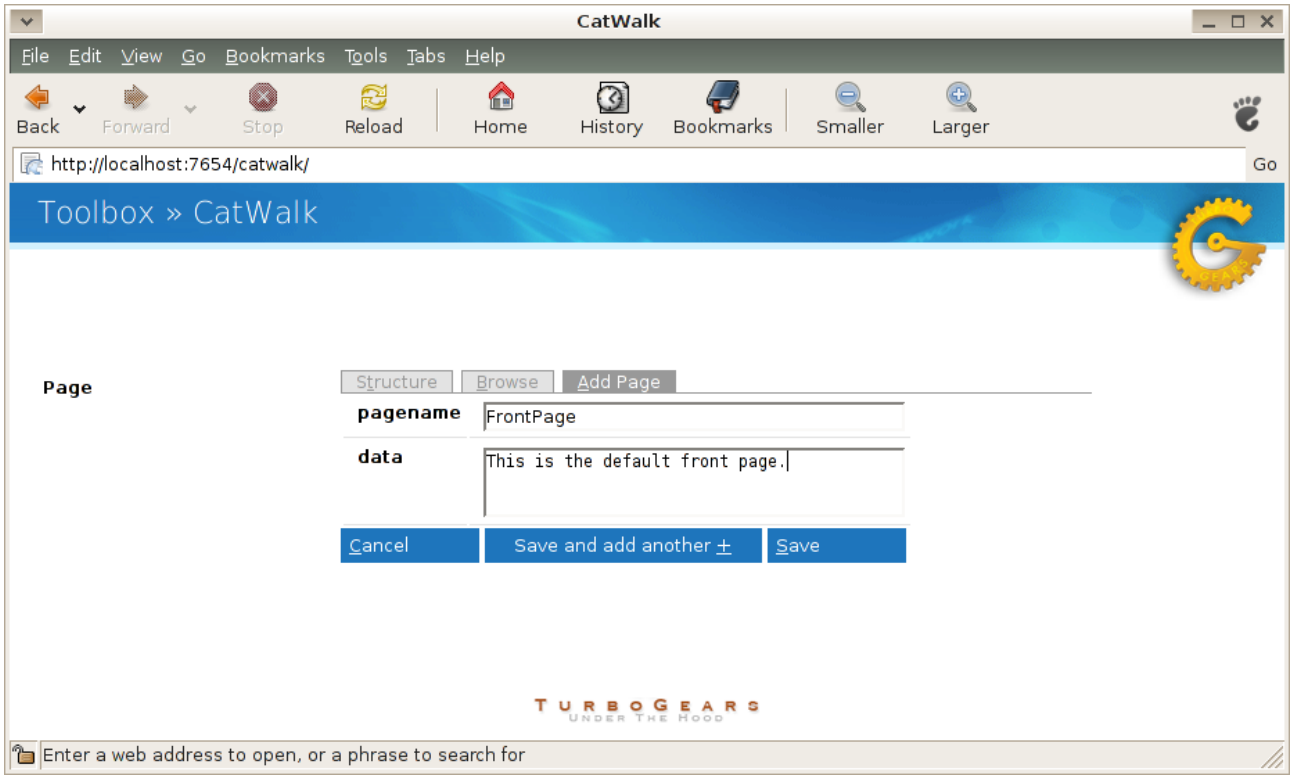

Fig. 5.14 TuboGears Toolbox : Catwalk – data entry in a page

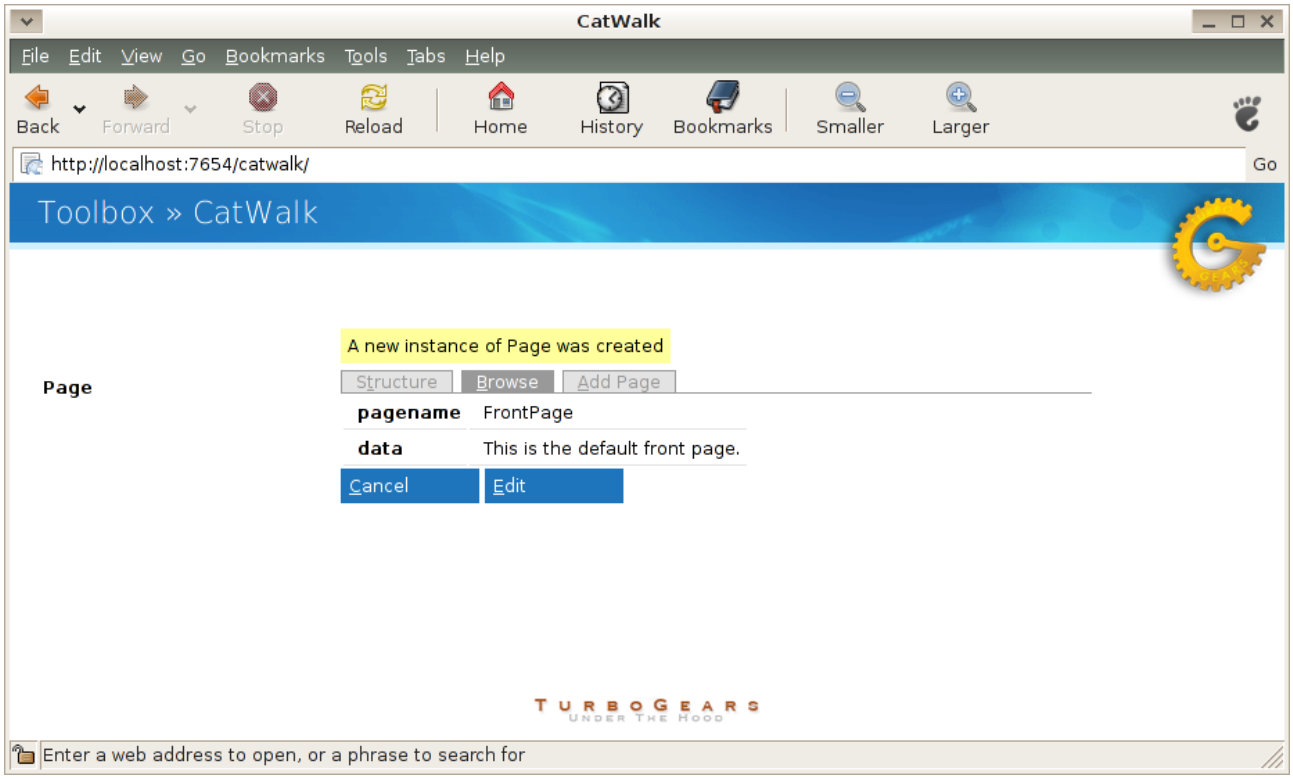

Fig. 5.15 TuboGears Toolbox : Catwalk – new instance of a page

Now http://localhost:8080 will display the newly created page (Fig. 5.16).

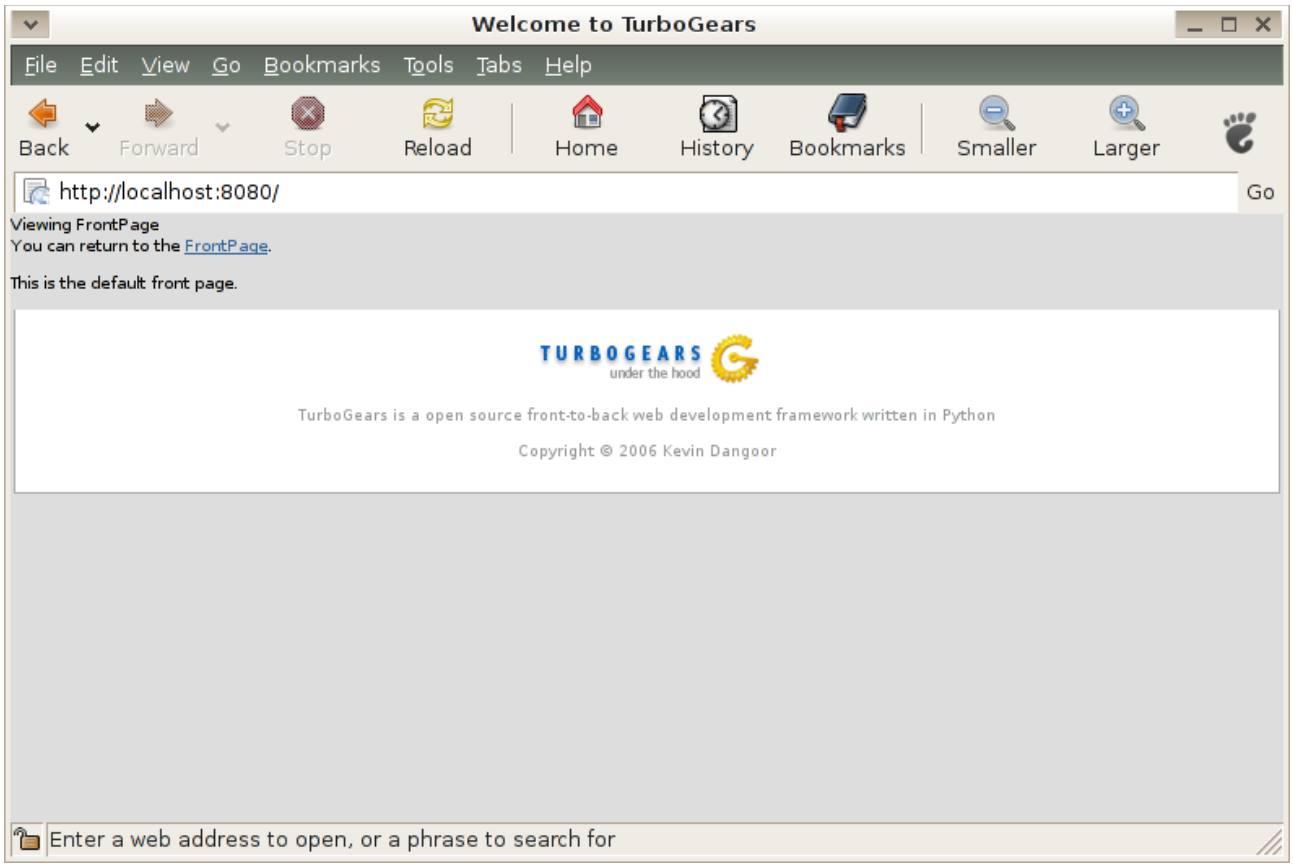

Fig. 5.16 TuboGears Toolbox : newly crated front page

Editable Pages.

Wiki pages can be edited just by clicking a "Edit This Page" link. To provide this functionality a new template for editing has to be created.

```
cd wiki20/templates
cp page.kid edit.kid
cd ../..
```

In edit.kid, change the text at the top from "Viewing" to "Editing" and replace the for the data with:

```
<form action="save" method="post">
   <input type="hidden" name="pagename" value="${page.pagename}"/>
   <textarea name="data" py:content="page.data" rows="10" cols="60"/>
   <input type="submit" name="submit" value="Save"/>
</form>
```

To use this template now, an "edit" method is required in the controller:

```
@expose("wiki20.templates.edit")
def edit(self, pagename):
    page = Page.byPagename(pagename)
    return dict(page=page)
```

This method is very similar to the index method. The main difference is that the index method renders the wiki content as HTML, and this one returns the straight text. Now to get access to this method, the welcome page has to point to this method. At the bottom of page.kid, we add:

```
<p><a href="${tg.url('/edit', pagename=page.pagename)}">Edit this page</a></p>
```

Now on reloading the browser page, the edit link will be visible and functioning but saving the text has not yet been implemented (Fig. 5.17).

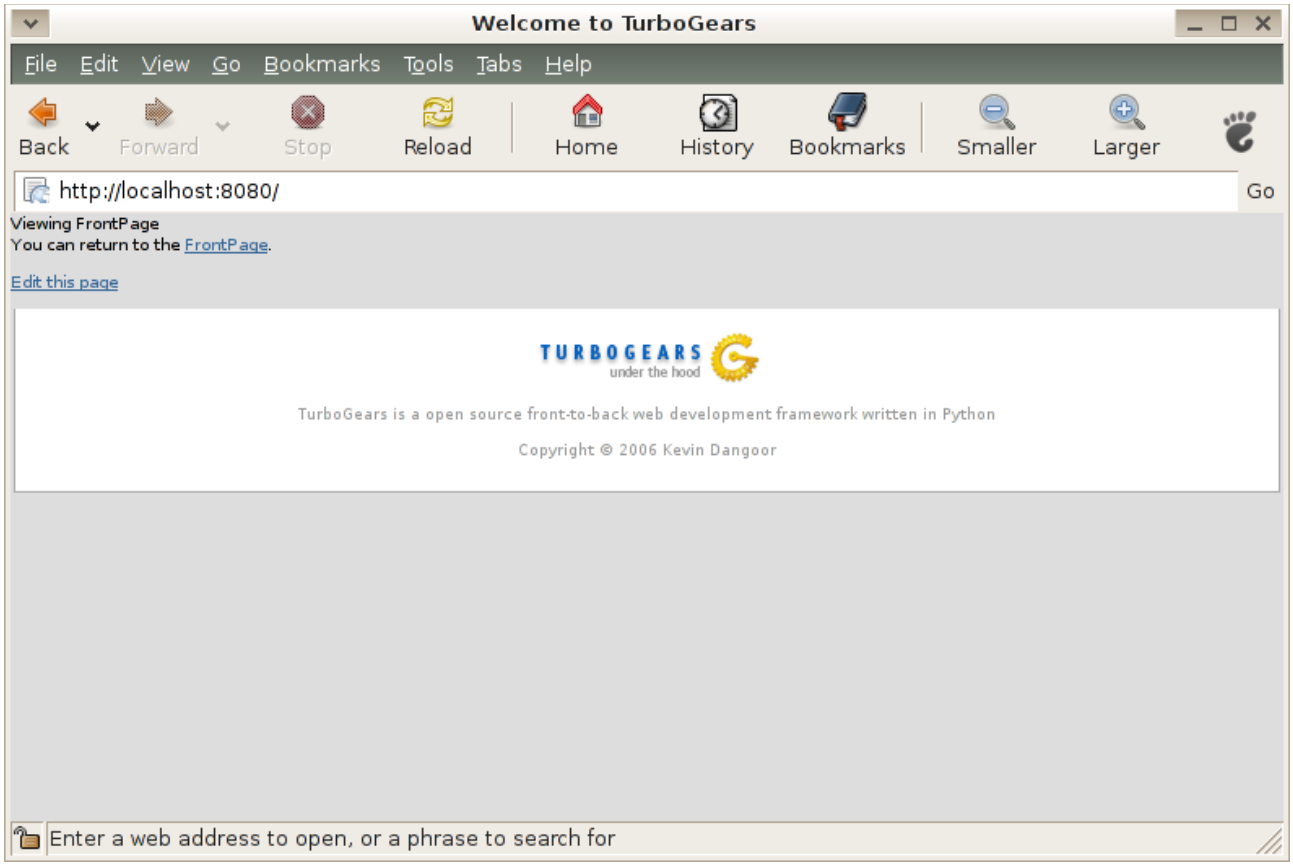

Fig. 5.17 TuboGears Toolbox : Catwalk – reloaded browser with edit link

Saving the edits

To provide the save functionality, we need to make a method called save in our controller. Here's what it looks like:

```
@expose()
def save(self, pagename, data, submit):
    page = Page.byPagename(pagename)
    page.data = data
    turbogears.flash("Changes saved!")
    raise turbogears.redirect("/", pagename=pagename)
```

Interesting things to note about the outcome of this operation which results in an output as given in Fig. 5.18, are:

1. Unlike the previous methods, this one is just exposed without any template specified. That's because it will only redirect the user back to the viewing page.
2. A transaction is set up implicitly, if the database supports transactions.
3. The "page.data = data" is all it takes to run the UPDATE SQL statement.
4. The turbogears.flash() call is setting a notification message to appear in the user's browser on the next screen. The "tg_flash" variable is referenced in the master.kid template.
5. Since an redirect is raised as an exception, all other processing is short circuited.

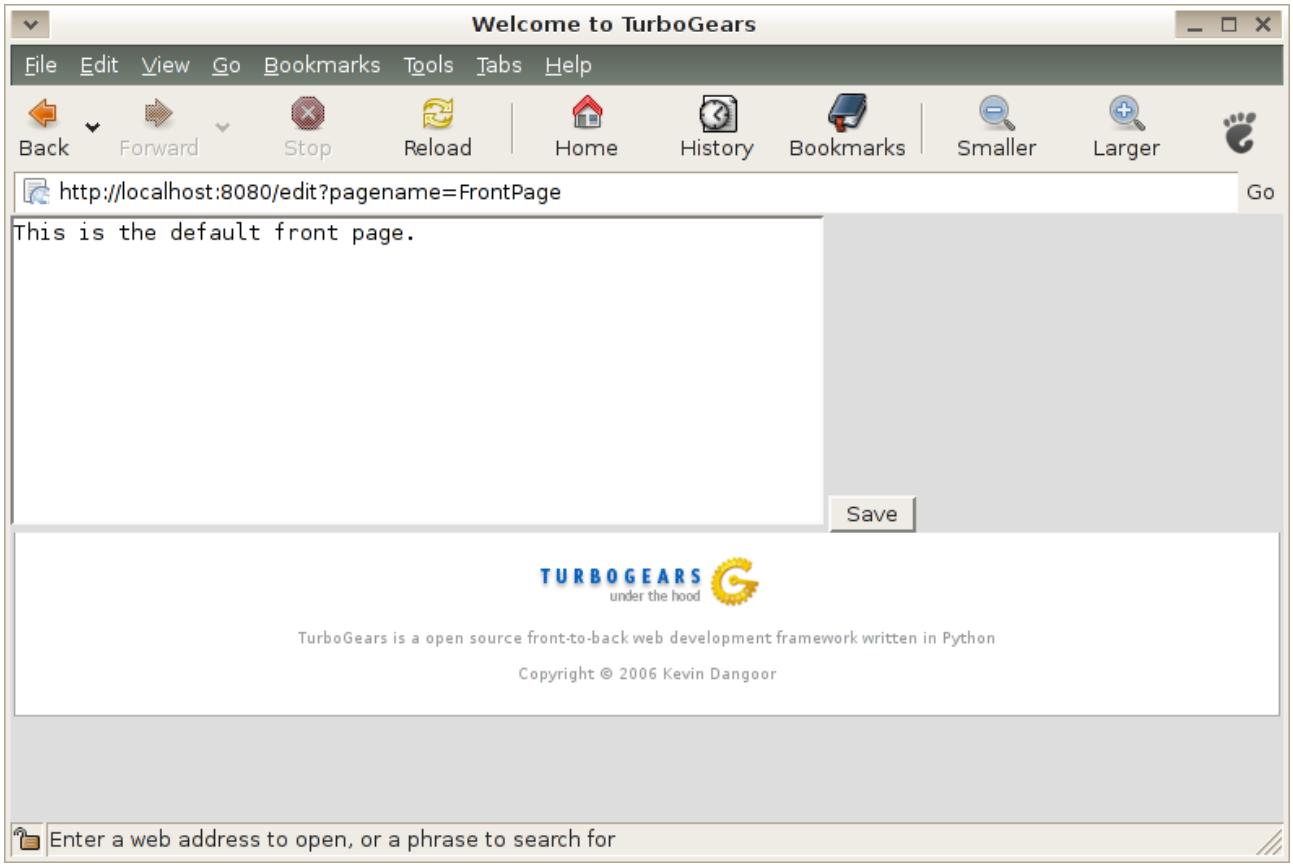

Fig. 5.18 TuboGears Toolbox : saved page

Now the full save functionality is present and can be tested (Fig. 5.19).

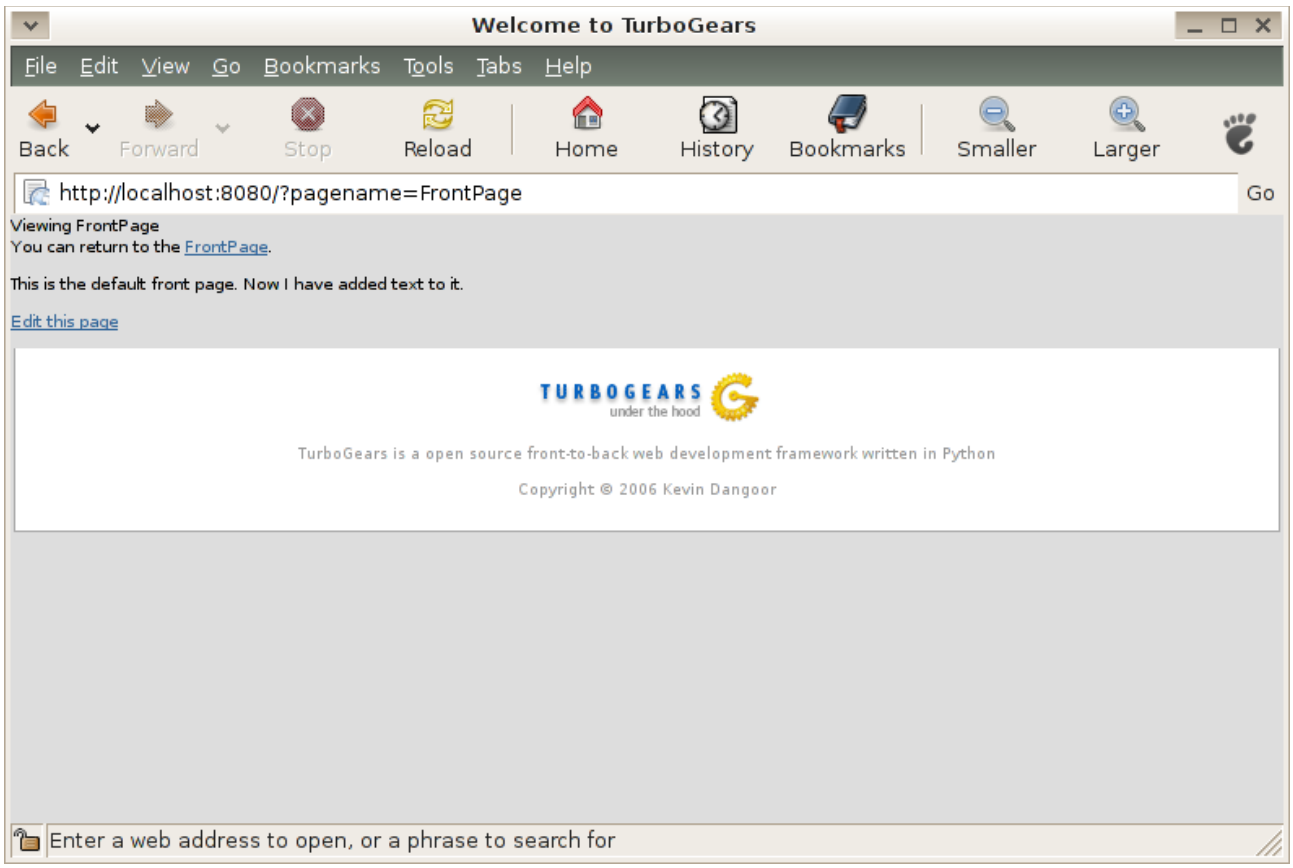

Fig. 5.19 TuboGears Toolbox : saved page for testing

Friendlier URLs

To change the URLs to user friendly ones ie. from "/?pagename=Foo" to "/Foo" a default method that CherryPy will call whenever no other method matches can be written:

```
@expose()
def default(self, pagename):
    return self.index(pagename)
```

And now, we can change the HTTPRedirect in the save method to:

```
raise turbogears.redirect(turbogears.url("/%s" % pagename))
```

WikiWords

WikiWords have also been described as WordsSmashedTogether. A typical wiki will automatically create links for these words when it finds them. This can be implemented using a regular expression. Start by adding an import to our controller (Fig. 5.20) :

```
import re
```

A WikiWord is a word that starts with an uppercase letter, has a collection of lowercase letters and numbers followed by another uppercase letter and more letters and numbers. Here's a regular expression that meets that requirement:

```
wikiwords = re.compile(r"\b([A-Z]\w+[A-Z]+\w+)")
```

Put that line just above the Root controller class. Then, we need to use this regex. Just below the publish_parts line of the index method (that's the one that generates the HTML), add this:

```
root = str(turbogears.url("/"))
content = wikiwords.sub(r'<a href="%s\1">\1</a>' % root, content)
```

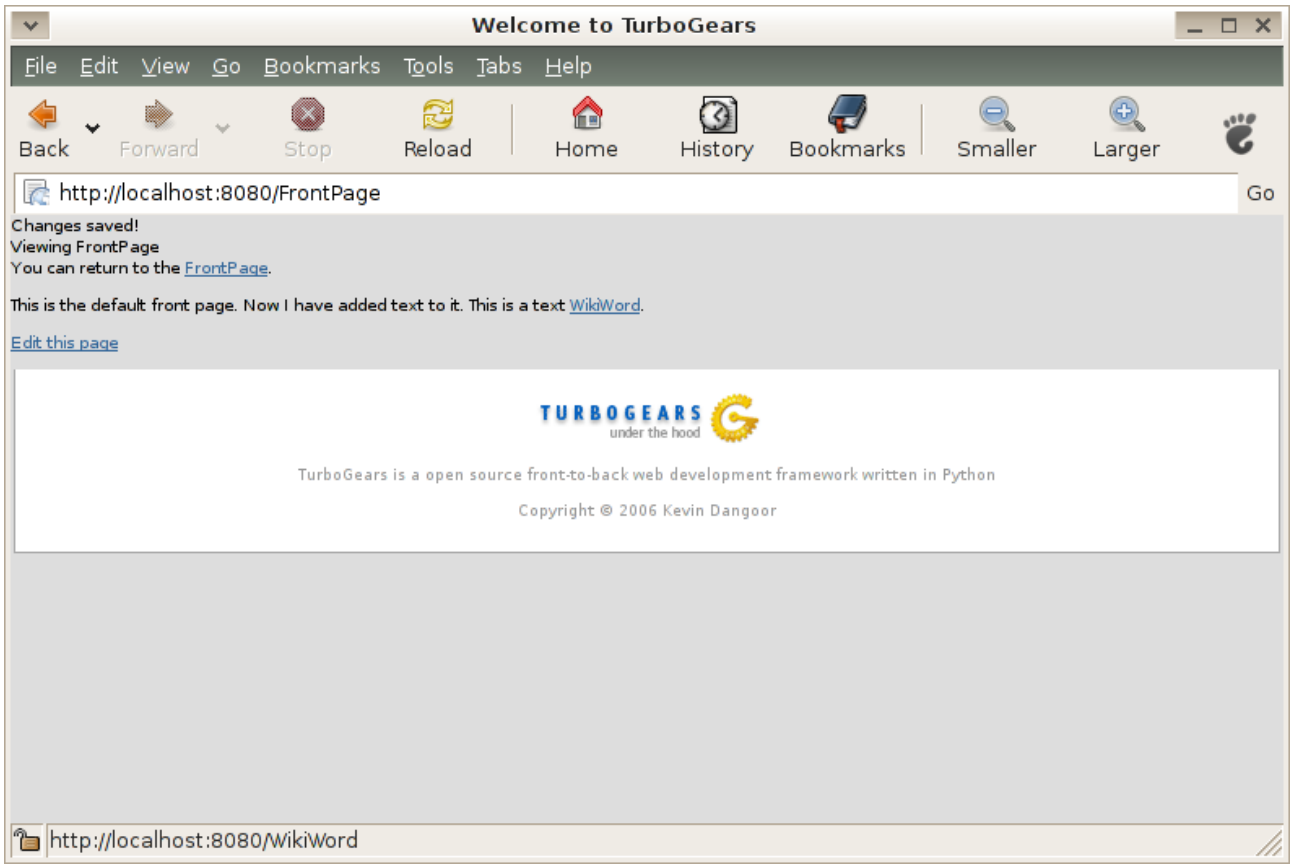

Fig. 5.20 TuboGears Toolbox : page with WikiWord functionality

Now the WikiWord functionality is active (Fig. 5.21).

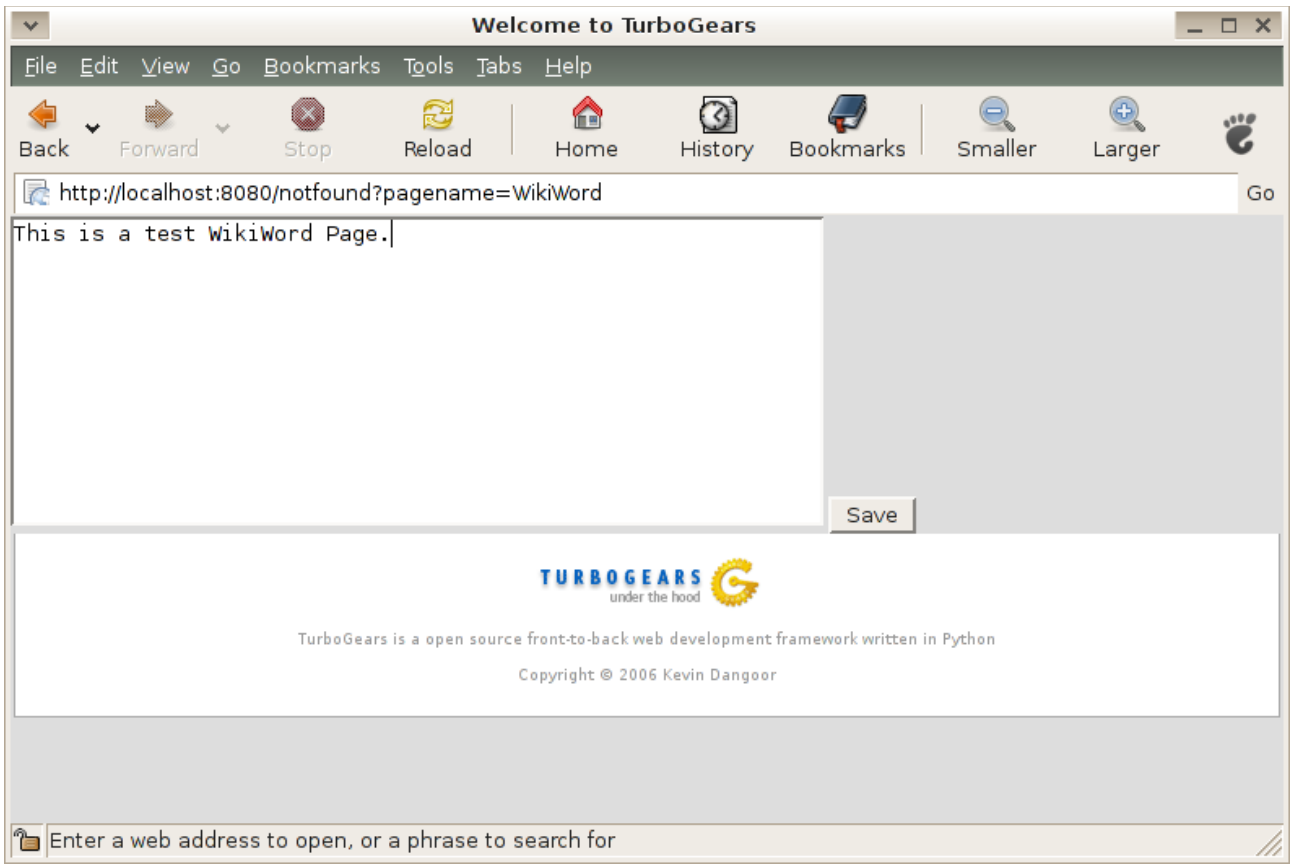

Fig. 5.21 TuboGears Toolbox : test WikiWord page

The changes made vide input entries can be saved and the message in this regard will be displayed (Fig. 5.22).

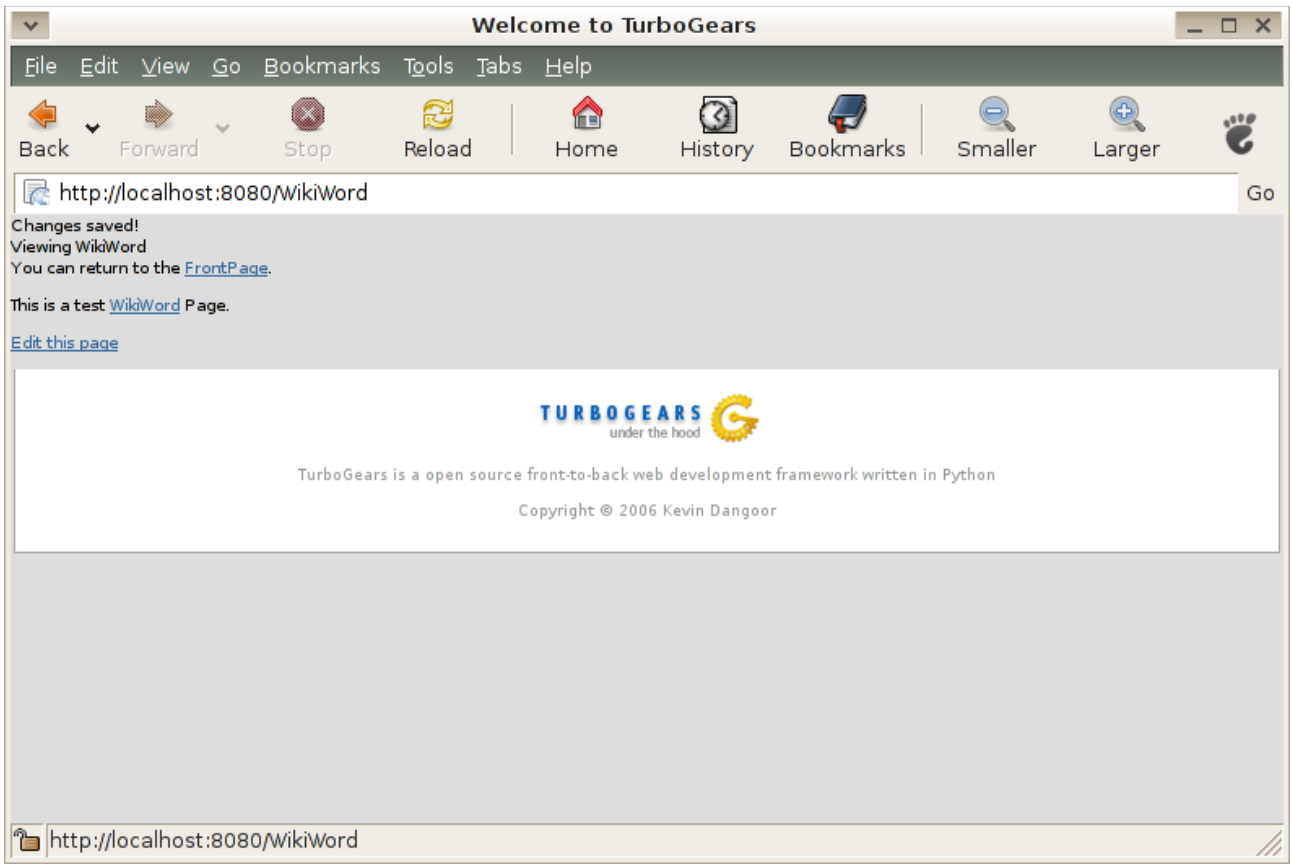

Fig. 5.22 TuboGears Toolbox : saving changes

Missing Pages

To check for pages which do not exist the following simple approach is used: if the page doesn't exist, the edit page is to be displayed. In the index method, we'll check to see if the page exists. If it doesn't, we'll redirect to a new notfound method. Change the index method like this:

```
try:
    page = Page.byPagename(pagename)
except SQLObjectNotFound:
    raise redirect("/notfound", pagename = pagename)
```

We'll need to import the exception as well. Add this to the top of the file:

```
from sqlobject import SQLObjectNotFound
```

And, we'll add the not found method. We'll just reuse the edit template:

```
@expose("wiki20.templates.edit")
def notfound(self, pagename):
    page = Page(pagename=pagename, data="")
    return dict(page=page)
```

With SQLObject, just instantiating an object is enough to insert it in the database, so this method will create a brand new page in the database.

Adding a page list

Most wikis have a feature that lets the user view an index of the pages. A new template, pagelist.kid with the page.kid as a starter is used:

```
cd wiki20/templates
cp page.kid pagelist.kid
cd ../..
```

The body of the pagelist.kid template is changed to look like this:

```
<body>
   <h1>All Of Your Pages</h1>
   <ul>
      <li py:for="pagename in pages"><a href="${tg.url('/' + pagename)}"
py:content="pagename">Page Name Here.</a></li>
   </ul>
</body>
```

This is the first use of py:for, and it is straightforward. The li element will be repeated for each iteration of the for loop, and the for loop is specified just as it is in Python.

Let's add a link at the bottom of the master.kid template to get to the complete list of pages:

```
<p>View the <a href="${tg.url('/pagelist')}">complete list of pages.</a></p>
```

Since we're referring to a "pagelist" method, we should probably create it:

```
@expose("wiki20.templates.pagelist")
def pagelist(self):
   pages = [page.pagename for page in Page.select(orderBy=Page.q.pagename)]
   return dict(pages=pages)
```

This is the first use of Page.select. Here all of the Page objects from the database are selected and ordered by pagename. The "Page.q.pagename" represents the "pagename" attribute as a query parameter. Page.select returns a SelectResults object which you can iterate over, making the one liner above easy.

After the modifications, the default front page will be as in Fig. 5.23

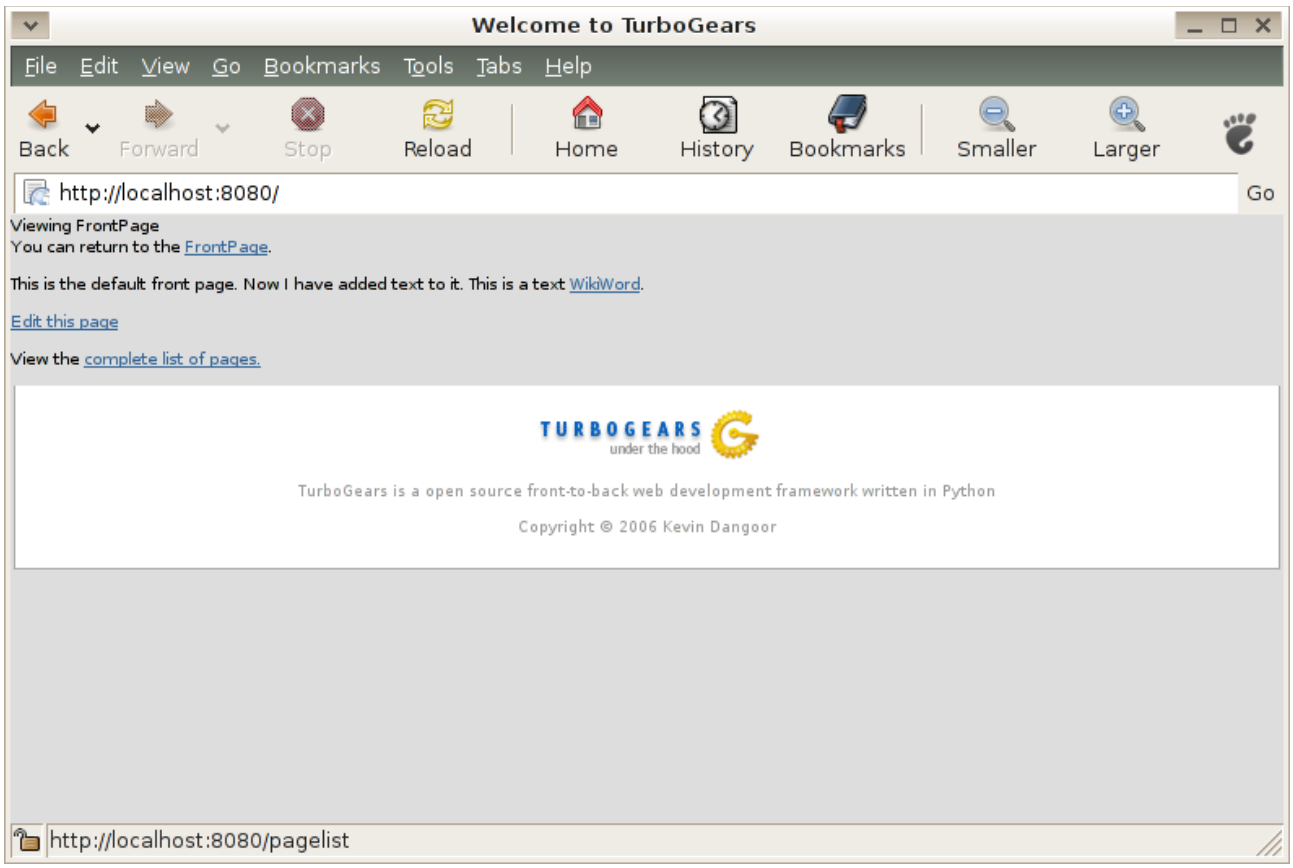

Fig. 5.23 TuboGears Toolbox : default front page after modifications

The list of pages can be views as in Fig. 5.24.

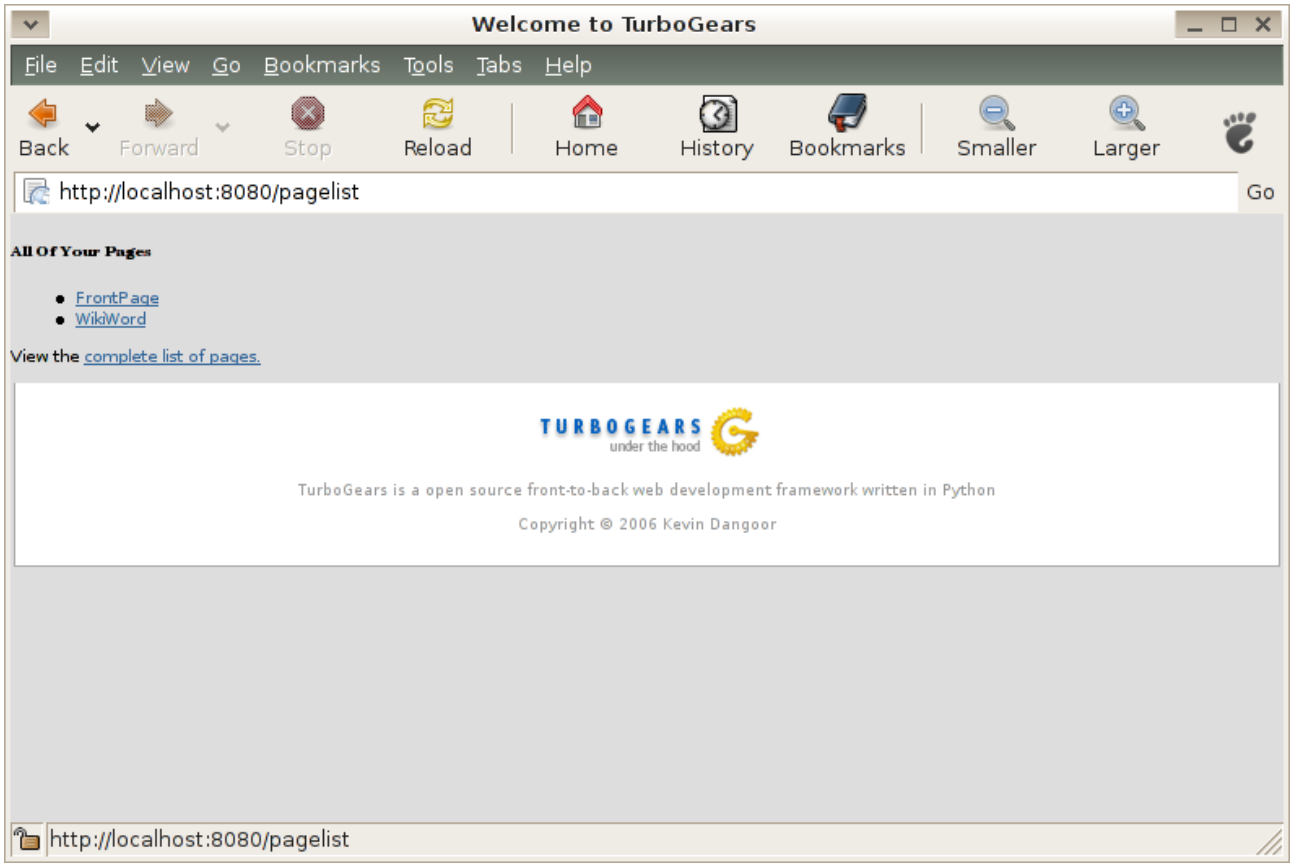

Fig. 5.24 TuboGears Toolbox : viewing list of pages

Finally after adding a few more pages, the links to all pages are visible as in Fig. 5.25.

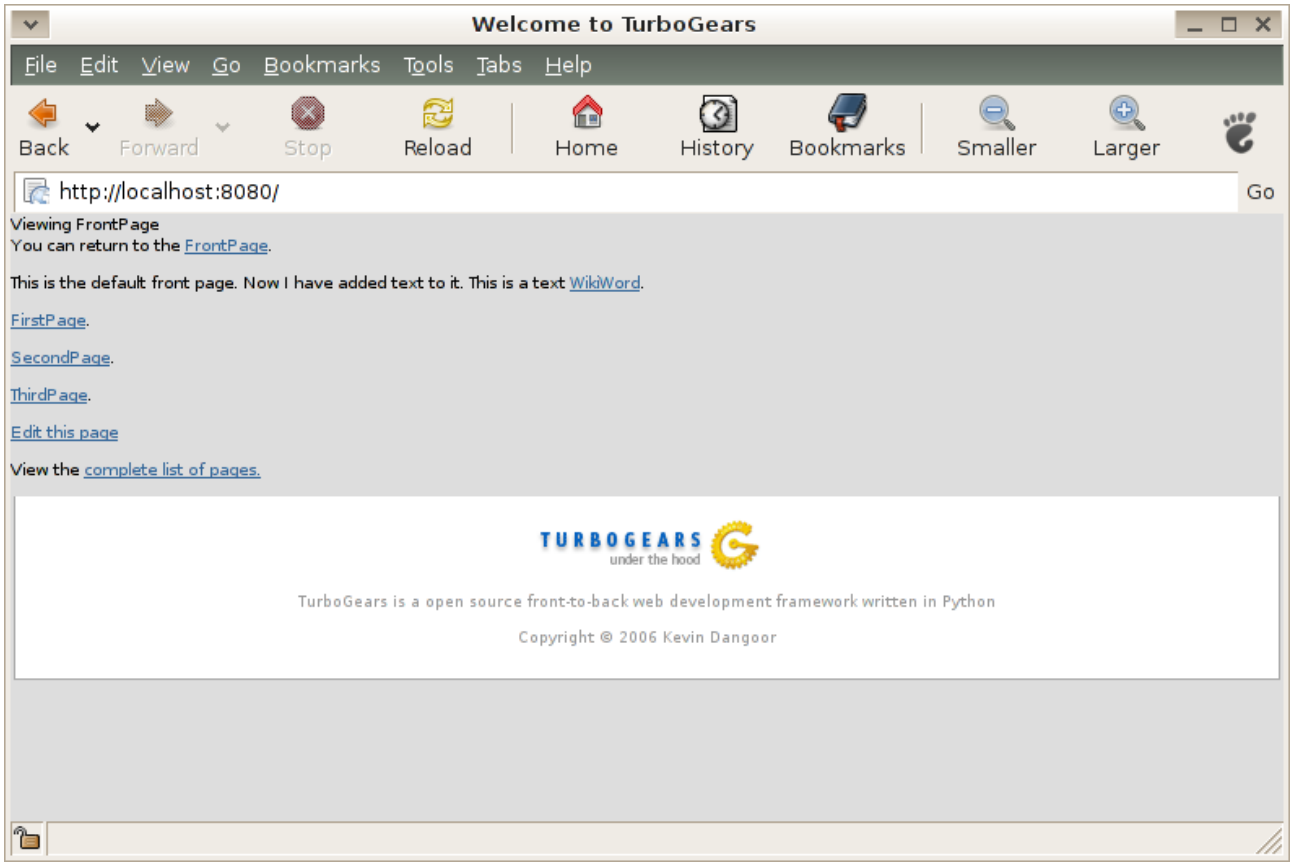

Fig. 5.25 TuboGears Toolbox : links for all pages

AJAX using JSON instead of XML

AJAX was discussed earlier in Section 4.4. In this section, we will discuss AJAX using Java Script Object Notation (JSON) instead of XML. JSON is easy and lightweight and efficient to use for all browsers. To use JSON for this TurboGears example, we just have to tell TurboGears that we want to use it. Just below the expose decorator for pagelist, let's add another:

```
@expose("json")
```

The location http://localhost:8080/pagelist?tg_format=json has the pagelist in JSON format.

JavaScript will be written using MochiKit. The "view complete page list" link will be changed to pull in the pagelist and include it right in the page currently being viewed. All changes will all be in master.kid. MochiKit chould be included in all of the pages. This can be done right from the project configuration file. This file is config/app.cfg in the Python package that quickstart created. In this file there will be a commented out "VIEW" part of the config. Uncomment the line that says "tg.mochikit_all" and change the value to True. The line will end up looking like this:

```
tg.include_widgets = ['turbogears.mochikit']
```

This setting will tell TurboGears to include MochiKit in every page. After making this configuration change, **the server has to be restarted.**

Next, the page list link is replaced with one that will invoke a JavaScript function. We'll also add an empty container for our pagelist:

```
<p>View the <a href="#" onclick="requestPageList()">complete list of pages.</a></p>
<div id="pagelist">
</div>
```

Now, we use MochiKit to retrieve the page list in requestPageList, which we'll define in a script tag in the head of the page:

```
<script type="text/javascript">
   function requestPageList() {
   var d = loadJSONDoc("${std.url('/pagelist', tg_format='json')}");
   d.addCallback(showPageList);
   }
```

MochiKit includes a function for doing an asynchronous XMLHttpRequest and converting the result from JSON into a JavaScript object.

loadJSONDoc returns a "Deferred" object. The idea with a Deferred is that we know that our request for the pagelist will happen *some time in the future*, but we don't know when. A Deferred is a placeholder that allows us to specify what happens when the result comes in. We have a very simple requirement here: call a function called "showPageList", which we'll write now:

```
function showPageList(result) {
   var currentpagelist = UL(null, map(row_display, result["pages"]));
   replaceChildNodes("pagelist", currentpagelist);
}
```

When loadJSONDoc gets its result, it will pass it along to show PageList.

MochiKit.DOM allows the user to create new document elements by nesting tags in the JavaScript. We're creating a UL element, just like we had in our freestanding pagelist page. The first parameter is a mapping of attributes for the element, which is null in this case. The other parameters will all become child nodes. MochiKit's map function works just like Python's. For each element in result["pages"], we're going to call a function called row_display. The last part is to replace the contents of our with the id of pagelist with the new DOM elements we've just created.

Let's create our row_display function and close off our script tag:

```
function row_display(pagename) {
   return LI(null, A({"href" : "${std.url('/')}" + pagename}, pagename))
}
</script>
```

Each pagename from the pages list will get passed into row_display. We create an LI element (no attributes), and an A element with an href attribute. The A element also has the pagename as its

contents.

Now the page list link ***front page*** will give the output in the page itself.

This chapter begins with a brief note on the multi-tier application architectures, and moves on to discuss the Model-View-Controller (MVC) architecture in some detail. Business Logic implementation using Enterprise Java Beans or web services were discussed. From among the frameworks that MVC uses, Servlet and JavaBean have been discussed. Struts and turbogears which are the frameworks used by MVC were also studied.

FOSS ELECTIVE-2

UNIT 1: ***F/OSS and Enterprise Applications***

Website:

http://wiki.go-opensourse.org
http://www.utoronto.ca/ian/talks/
http://www.w3.org
http;//www.openldap.org

UNIT 2:*FOSS Development Methodology &Tools*

website:

http://www.tutos.org
http://opensource.mit.edu/papers/mockusapache.pdf
http;//gnuwin.epf1.ch/articles/en/cathedralbazaar/cathedral-bazaar.pdf
http://www.nongnu.org/cvs
http://gnu.org/software/make/manual/html-chapter/make.html

References:

->J Feller and B Fitzgerald.Understanding Open Source Software Development.Addison Wesley 2002

->Fielding RT.Shared leadership in the Apache project.CACM,42(4), April 1999.

->A Mockus,RT Fielding,and J Herbsled.A case study of Open Source development:The Apache Server.ICSE'2000.

UNIT 3:*Presentation Layer*

Website:

http://developer..mozilla.org/en/docs/AJAX
http://www-128.ibm.com/developerworks/web/library/wa-ajaxintro1.html
http;//w3.org/DOM
http://developer.mozilla.org

UNIT 4:*Datastore Layer*

website:

http://techdocs.postgresql.org

http;//www.varlena.com/GeneralBits
http://www.pgsql.ru/db/pgsearch
http://www.postgresql.org/docs/books/awbook.html
http;//www.commandprompt.com/ppbook/
http://www.tricknology.org/foilware/

UNIT 5:*Business Logic Layer*

website:

http://java.sun.com/blueprints/patterns/FrontCont roller.html
http://java.sun.com/blueprints/patterns/MVC.html
http://struts.apache.org/
http://www.turbogears.org/about/index.html
http://www.sqlobject.org
http://www.cherrypy.org
http://kid.lesscode.org
http://www.mochikit.com