A Front-End For Knowledge 2.0 Platform

Master of Science Thesis
Advanced Internet Applications

By
Amir Ghaffari

Supervised by
Dr. Hans Wolfgang Loidl

School of Mathematical and Computer Science
Heriot Watt University

August 2011

DECLARATION

I confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed:

Date:

Abstract

Academic research is considered as a foundation resource for more investigation in any field. Due to this brilliant role many companies and government make huge investment in community of scholar for academic research. Importance of academic research can be a good reason for considering more on their publication and dissemination methods. Earlier, publishing was strictly related to paper.

Today paperless publishing or electronic publishing is increasingly popular in scientific articles. Because online publishing is faster than a print one and its cost is lower in production and dissemination stages, a number of journals have moved entirely to the electronic publication. E-publishing in which books, journals and magazine are being produced and stored electronically rather than in print (1).

In particular Web 2.0 provides new features to further enhance web publishing. Web 2.0 brings wide range of features such as searching, linking, cross referencing, online referee, rating analysis. It causes considerable increase in number of scholarly journals on the Web due to quality of presentation, convenience and low cost.

This project develops a Front-End base on web 2.0 and RIA technologies to provide an easy access to data and desktop functionality for knowledge platform. Front-End have used web 2.0 features for increasing user collaboration. These features are rating, commenting, tagging, searching and linking that makes user's collaboration much more effective in knowledge dissemination process. Moreover, Front-End provide an easy to use and secure peer review procedure that is another aspect of knowledge dissemination. Front-End have tried to introduce a new concept that reduces strict aspect of scholarly publication by providing a more user-friendly and collaborative environment.

Front-End aims to provide an interactive and collaborative online platform for both knowledge producers (Author) and knowledge consumer (Reader).

Acknowledgements

First of all, I would like to thank my supervisor, Dr. Hans Wolfgang Loidl, for his proactive approach and support in getting this thesis done as soon as possible.

I also wish to express my Appreciation to Professor Phil Trinder for the help and guidance he has given me during master course.

Finally, I would like to express special thanks to my wife, Shiva who stood beside me and encouraged me constantly.

Table of Contents

Chapter 1

Introduction

Section 1.1 explains the research context in academic knowledge dissemination in which the project is developed. Section 1.2 gives the objectives and tasks. Section 1.3 explains the organization, structure and requirements of the dissertation.

1.1 Context

The academic world places strong emphasis in research and subsequent publications of their findings. The academic community is seeking to make the exchange of research's result faster and better, at reasonable cost. Traditional publishing can be defined as a text, pictures or images on paper in form of books, newspapers etc. It was strictly related to paper. On the other hand, in electronic publication books and journals are produced and delivered to consumers in digital formats rather than in print. These publications have all qualities of the normal publishing like the use of color, graphics, and images and are much convenient also. To respond to that demand and give users always more flexibility many standards for electronic publication such as OEB and PDF for e-books have been developed. Electronic Ink other technology that can have an important role on the e-publishing and some companies such as E-Ink Corporation and Xerox have invested heavily in it.

Information by means of e-publishing can be provided online through the Internet, by email publishing (newsletter publishing), web publishing (HTML, XML) or on disk or CD-ROMs etc. Popularity of web publishing as compared to other technologies has certainly grown in recent years and during the last decade web technology play a main role as a publisher. Online publishing that known as web publishing can be defined as a distributed electronic format of information. Web is an important resource for scientist and researcher and offers a wide variety of possibilities for communication between scholars. Before 1999, Web 1.0

technology was used for web publishing and user's role was limited only to reading the information. There was no active communication and support only static websites with read passively.

Open access means no charge and without license restrictions in scientific literature, has become a demand by many people and emerged into an international movement these days. The idea is supported through a growing number of online journals that publish scientific publication in electronic form and often freely (2).

Web 2.0 brings more user participation by supporting dynamic content, interactive and up to date content and flexibility to web publishing. It is more interactive in contrast to websites where users are limited to the passive viewing of knowledge content that was created for them. For example usage of web 2.0 technologies in social networking offer fundamental shift in the way people communicate. Marketers also have used Web 2.0 tools to collaborate with consumers on product and service promotion.

In the same way web 2.0 allow author to communicate with publisher during the process of publishing manuscripts. Prepublication review process of academic papers can be done electronically. Referee's comments will be shared anonymously with the author and provide impression of the article and specific suggestions for the article's improvement.

Publication ranking and quality assessment of research is a growing trend in many countries. The past decades have witnessed a growing competition among individual scholars, universities, and journals to achieve high rankings. For constructing journal ranking we can use a number of active experts to assess the articles by their quality. In contrast there is a technique for developing ranking lists based on effect of citation on other articles. Computers can do the formula base method of ranking efficiently (3).

Nowadays, Authors prefer to publish their manuscript online because readers focus on retrieving information from the Internet and library try to deliver information to members by using electronic archiving. Also when journal are available to web all over the world can read the article. Web publishing is faster than a print one and not need to wait for print. Moreover, online referee process can be done as soon as possible. Linking is an online facility than can help readers to refer to other cited article by clicking a button. Due to a first-mention advantage a growing number of online publishers and publishing services have emerged. Publishers try to handling of electronic manuscripts, copyediting (words in manuscript is ready to publish), formatting (format that publication must be published), typography (preparing written material for publish) by using web technology and the production of a suitable version for delivery to the end users (4).

1.2 Objectives and Aims

The project aim is using the web 2.0 technological tools to design and implementation a Front-End for a scientific publication platform to ensure effective and accurate communication on publication by supporting the submitting publication file and appropriate meta data, editorial board, double blind referee, ranking and role base user management. Front-End provides an easy access to data available in the infra-structure. It has used Rich Internet Application (RIA) technologies to offer a rich and desktop functionality to improve user satisfaction. For support RIA capability Front-End has applied new web 2.0 technologies that are listed below:

- Ajax: For doing delete, insert and update, Front-End has applied Ajax technique to exchange data with a server and update parts of web pages without reloading the whole page.
- JQuery to apply client side validation and confirmation: In Front-End JavaScript and JQuery that are appropriate techniques for client side validation and conformation are used.

Front-End aims to mix server side technologies (J2EE) with RIA technologies (platform independent ones) to provide web 2.0 features (rating, commenting, tagging, linking, etc) for Knowledge platform.

This master thesis constitute of two main tasks:

1.2.1 Theoretical

Theoretical part is a study of the characteristics of the scientific knowledge platform. Project aims is provide improvements in functionality, usability, availability, efficiency and cost in the access of scientific and scholarly knowledge. One main focus will be on the use of web 2.0 technologies for the electronic dissemination of information. This part aim is to understand the complete publishing process and finding roles of web 2.0 technology for use of rich internet application in a modern electronic publisher. Common technologies are currently available in RIA. We will study how to adapt RIA in the user interface (UI) layer and enhance user interaction. Also sophisticated server interaction mechanism will be discussed.

1.2.2 Practical

This part consists in implementation of a Front-End to web knowledge platform by using web 2.0 technology tools. The goal of this part is design and implementation of the web publisher for scientific knowledge with interactive capabilities provided by web 2.0 technology. In order to achieve these aims, the project tries to investigate suitable tools and methods to be used in the project. By using RIA technologies in Front-End, we are going to restores the client's abilities to be more interactive like desktop applications. The main functionality that Front-End supports are listed follow:

- Submitting scientific papers, including meta-data
- Search for papers by fields in the meta-data
- Categorization of data by Tagging, Keywords and Tag Cloud
- Increasing user participation by Commenting and Rating
- Using rich internet application(RIA) technique such as Ajax, JQuery and JavaScript to improve desktop functionality and decreasing server overhead

1.3 Requirements

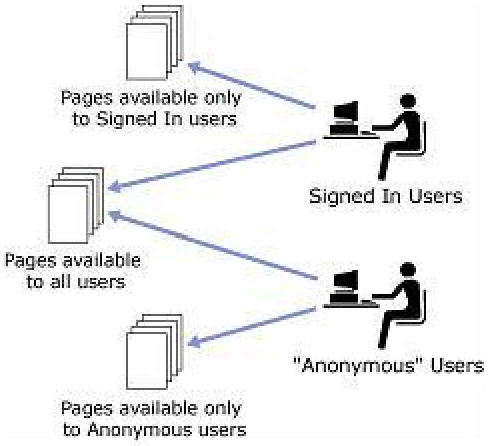
Front-End's specification and technical requirement are listed follow:

1.3.1 Role-Based Security Mechanism

Only privileged users are permitted to perform certain operations on the infrastructure. One of the most convenient security mechanisms is role-based design because access granted at the role level and then it allows individual users to be assigned into roles. System security service should check two different stages:

Authentication: This service is allowed when the user needs to be authenticated using a username and password.

Authorization: This service check whether user's role are allowed to access to a particular page.


Pages available only to Signed In users
Signed In Users
Pages available to all users
"Anonymous" Users
Pages available only to Anonymous users

Two different roles have access to different resource

1.3.2 Search for Papers

Providing a search interface which allows different kind of publication such as articles, papers, electronic books to be found is an important feature of Front-End. Knowledge platform has the potential to store huge amount of information because digital information does not have physical space limitation to contain it. Search page should make enable users to search publication by title or author name. Publication's keywords and user's tags are other options that supporting an advanced search base on them can increase user satisfaction.

1.3.3 Upload Publication and Supplementary Data

Since one of the main aims of Front-End is providing a safe and easy to use mechanism for uploading information. In Knowledge Platform the producer of information are authors. Due to this prominent role, Front-End should provide a way for authors to upload their publications and its supplementary files. The selected method should be secure and optimize.

1.3.4 Double/Single-Blind Refereeing

Well-functioning editorial and publishing process are essential part of publishing. In **single-blind** reviewed reviewers know the author's identity, but not vice versa. Blinding the identity of reviewers is useful because referee can comment freely. Another method is **double-blind** review, whereby both referee and author remain anonymous and the identities of the author and referees are both hidden. This feature avoids all potential bias and makes it easier for reviewers to focus on the paper itself. Front-End aims to provide peer review process for reader and authors.

1.3.5 Cross-referencing

Cross-reference means reference from one part of a book, text, index or file to another part containing related information. It is an instance within a document which refers to related or synonymous information elsewhere.

1.3.6 Ranking

Front-End should provide a method to get feedback from reader. Ranking publication based on reader's interest can provide a good factor for providing a list of top rated publication. Rating is one of the important features of Web 2.0 technologies.

1.3.7 Web 2.0 Features

Since web 2.0 technologies bring a new possibility to user for searching information base on their own categorization. Front-End should apply new technologies to provide a method for user categorization of data. This can be done with using new concepts such as tagging and tag cloud.

1.4 Outline

The thesis report is divided into six chapters:

1) Chapter two contains literature review. It describe the evolution, presents background information on web publishing and main process of web publishing in order to gain more understanding about required feature in modern and suitable platform among the various platforms. This section also reviews the different tools and programming languages that can be used during development process of Front-End base on web 2.0 technologies.
2) Chapter three focus on design and implementation. In this chapter, the stages of analyze, design and implementing is explained. It discusses more on using tools and technologies that is used for design and application implementation. Architectural pattern is discussed. It presents the pattern and technique that is used to isolate data model, business rules and user interface. It will go in detail how interaction between layers is done.
3) Chapter four is evaluation. Evaluation plan of the project is described. In addition, the evaluation process and the result are discussed in this chapter.
4) Chapter five is conclusion. It includes a discussion of the achievements, limitations and future work.

Chapter 2

Literature Survey

This chapter presents a survey of literature on scientific knowledge platform. First, the literature review will go through the main objectives for developing knowledge platform and why they are important. Then it will investigate previous research and discuss the fundamental process that must be done in Front-End part of scholarly publication. Main features and roles that are involved in Front-End of the scientific knowledge dissemination process will be discussed. Advantages and disadvantages of current technology that can be applied during development of web publication system will be explained. Fundamental parts of web 2.0 technologies such as Ajax, Rich Internet Application (RIA), jQuery and how these advances can be used during development processed will be explored. We will discuss about how web 2.0 techniques can improve the functionality of publication process by applying them in the Front-End.

2.1 Introduction

Scientific publication is an important subfield of publication. In recent decades considerable growth has occurred in academic publishing especially in developing countries as they have advanced in science and technology. Academic publication is published in form of article, book, presentation or thesis . Web knowledge platform is using the new technology and tools to develop a platform for dissemination scientific publication to ensure secure communication on publication by supporting the main steps in online publishing. Electronic publication has become increasingly interested to provide open access to scholarly and scientific research. Scholarly society increasingly finding that providing easy access in electronic format via the Web can be the most powerful and economical way for knowledge dissemination.

There are common issue that almost online publisher support them. Significant achievement in technologies and development tools has brought major changes in the way of design and implementation of web publishing. In scientific publication process there are some essential steps that must be taken before final dissemination. Consequently, there are main role in publication process such as author, referee, editor, etc that must be discovered and modeled in an online publishing. Clearly, in public knowledge sharing security has a vital role because people are able to access the system in all over the world. Recent dramatic changes to the web technology can be used efficiently to electronic publishing. More interactive web pages with capability such as asynchronous communication can be applied in web publication design and implementation. Access restriction to knowledge content and subscription are significant features that will be considered in online dissemination. Reading tools are also used to able user to view articles in common format such as HTML and PDF. These formats also are supported in most mobile devices such as iPhone.

2.2 A survey and evaluation of current web knowledge platform

In this section we are going to investigate previous research and platform to find features that they support. We will explore their advantage and disadvantage then we can choice the best solution for each part to use them during development process.

2.2.1 CiteSeerx

It is a scientific literature digital library and search engine that focuses on the literature in computer and information technology. Rather than creating a new digital library, it tries to provide algorithms, data, metadata, services, techniques, and software that can be useful for other digital libraries. It does not try to model the whole peer review and journal production process. CiteSeerx use new methods and algorithms to index PostScript and PDF research articles on the Web. Citeseerx use SeerSuite algorithm for creating academic search engines and digital libraries. Autonomous citation indexing (ACI) method helps to make automat the construction of citation indices (5). An ACI create a citation index from literature in electronic format automatically. It autonomously locates articles, extract and identifies citations to the same article that occur in different formats, and it also identifies the context of citations in the body of articles. CiteSeer download papers from the Web and then convert them to text. In next stage it parses the papers to extract the citations and the context in the body of the paper and store this information in a database. CiteSeer use heuristics to parses citation and extract fields such as title, author and year of publication. An ACI system help publishers by directing users to the journal's Web site.

CiteSeer use heuristics to extract fields such as title, author, year of publication and parses each citation using citation identifier like "[6]," "[Giles97]," in the document body. After extract the citations by using regular expressions try to analyze them (5). A helpful issue that ACI provides is allowing scientists to find work that cites their own work or is relevant to their research. Citation statistics are widely used for ranking. However, ranking based on citation statistics is not an accurate conclusion. Assumption that a large number of citations imply scholarly impact is not always true. CiteSeer give an opportunity to researchers to register to receive e-mail notification for new citations to papers of interest, or notification of new documents that match a personal profile.

2.2.2 Open access publication

An Open Access Publication is one that meets the following conditions:

The author grant to all users a free online right of access to read, download, copy, distribute and display the work publicly in any digital medium for any responsible purpose. It means to place a single copy of an work on the Web for anyone, anywhere to download. There is a cost to producing the article at the first, but no extra costs dissemination. In open access dissemination, people need to have access to computers and Internet, and minimum skills to take advantage of this freely available knowledge.

There are two strategies for open access publication:

1) Self-Archiving: scholars deposit their refereed journal articles in standards created by the Open Archives Initiative. Then search engines separate archives and users then need not know which archives exist or where they are located in order to find them.
2) Open-access Journals: These journals will no longer invoke copyright to restrict access to the material they publish. These new journals do not charge subscription or access fees and they find other methods for covering their expenses. Open access is possible because after the creation of the first digital copy of a work the cost of creating additional copies and distributing them on the Internet is marginal. In paper-based publishing open access is not possible due to paper base production costs, physical storage and distribution costs.

There are many platforms for disseminating Open Access content online and they have different performance. The main factor for performance is how well it allows a journal to perform in an online environment. Online dissemination makes it available anywhere in the world because internet connection in nowadays anywhere. Moreover, although Platform should be able to provide full-text, openly-accessible content, they should also include mechanisms for actively disseminating knowledge on the web in a secure ways.

Web is no longer a static HTML pages and hyperlinks. Web 2.0 technologies bring tools to provide more functionality for electronic publication. It provides tools to support further research and tools to promote access and discussion and interactive communication.

Open access publishing does not necessitate the use of open source tools but all tools we will discuss below are open source, although they have some difference in licenses. They are free to download and modification.

1) HyperJournal is an Open Source web application written in PHP. Features include (6):
    - HyperJournal automatically transforms cross-references contained in journal articles into bidirectional links that make user able to jump quickly to relevant article. For example by clicking on an author's name, system automatically searches search across database of linked HyperJournals and produces a citation list that includes all the articles written by the author or all the articles the author has cited or all the articles that cite the author.
    - HyperJournal Network connects all journals using the HyperJournal software.
    - It supports customizable interface, anonymous author, blind referee, popular file format.
    - Use of RDF metadata repository on the backend.
    - Editorial workflow is completely customizable

2) E-Publishing Toolkit is open source software for publishing open access journals. Developed in Python. The package is developed by the Living Reviews organization that is a scientific publisher. Most of the functionality provided by ePubTk can be described as offline abilities. Manage, create and maintain the content of scientific journals locally.
3) GAPworks is an online publication system developed by the German Academic Publishers (GAP) Project and it is funded by the German Research Foundation, DFG). GAPworks provides basic infrastructure for online publishing including a peer-reviewing process and a roles based user management, and other elements of the publishing process. It supports various types of publications that can be processed from submission to publication. It is developed on PostgreSQL database and is programmed in PHP.

4) DPubS (Digital Publishing System) is open-source software. It is developed to enable the organization, presentation, and delivery of scholarly publication such as journals, monographs, conference proceedings. It was developed within Cornell's Computer Science department in the early '90s and used for several years as the engine behind NCSTRL, a distributed network of Computer Science technical reports. The DPubS interface is based on a flexible XML and XSLT design and support full-text searching, Open Archives Initiative (OAI) compliance, flexible access controls, and e-commerce. It made provision for subscription services. It is developed base on Perl programming language.
5) Open Journal Systems (OJS) is developed by Public Knowledge Project (PKP), it is well supported by two major Canadian universities (University of British Columbia and Simon Fraser University). It is a journal management and publishing system currently being used by more than 8300 online journals around the world. OJS is a multi-journal publishing platform that supports refereed publishing process, from submissions through to online publication and indexing. It supports varying levels of article processing and full subscription options. It had the best comprehensive and clear documentation of any of the other discussed system.

   Following features are supported by OJS:
   - Online submission
   - Optional subscription module
   - Configurable reading Tools for content
   - Email notification
   - Commenting ability for readers
   - Online Help support
6) Topaz is a nonprofit organization related to the Public Library of Science (PLoS). It focused on developing of the Topaz open source software for collaboration to creation, management and sharing of information. It has a Service Oriented Architecture (SOA). They try to develop software framework to make easy development process of data driven applications

(Topaz platform). Ambra is other project that aimed at developing publication system that overlies the Topaz framework. Ambra is efficient system for the publication of quality-assured research in all areas of science. It is a web application based on Java that can be run compliant servlet containers.

Overview of Editorial Process in PLoS Genetics:

Submitted manuscripts are first reviewed by the Editorial board. Editorial board may decide to reject the paper or send it on to Associate Editors (AEs) for further review. The AE evaluates the paper and if it can pass this stage successfully then the paper is sent out for external peer review. Once the reviews have been received and considered by the editors, a decision letter to the corresponding author is drafted and sent. Once the reviews have been received and approved by the editors, an approved letter will be sent to author. The final decision can be one of the following options:

- Reject
- Major revision
- Minor revision
- Accept

7) Academic Archive Online(DiVA) is a common project between a number of universities in Scandinavia (7). The DiVA system developed and maintained at the Electronic Publishing Centre at Uppsala University, Sweden. The DiVA system started as a project in 2000 and has been in full operation since January 2003. It is now used by 15 universities in Sweden, Denmark and Norway and all of them are co-operating in development process. Today the archive contains mainly doctoral, undergraduate theses and research reports. It is based on Java and XML technologies and provides services such as harvesting via OAI-PMH. Many parts of the publication and archiving workflow are supported by XML. All metadata and document content, is stored as a DiVA Document Format in a uniform XML-based. Apache Lucene is used as the text search engine in DiVA, which gives a high performance searching process. All documents

published in DiVA system have a unique identifier that is assigned by using the Royal Library a URN:NBN resolution service.

Evaluations conclude: Most evaluated systems support the Open Archives Initiative Protocol Metadata Harvesting (OAI-PMH). Result shows the prominent role of interoperability in web publication and digital repository. Also modeling and implementation of scholarly publication is other main issue that all discussed systems support it in different level. Whilst every application has its own protocols but they all follow the same basic structure during peer review process.

There are some issue that are not covered in all evaluated systems such as role configurable (no hard coding for roles), code extensibility (such as plug-in, add-ins), Automated email alerts (for reader, author, editor, referee, etc).

There are three main requirements for each knowledge platform that we will discuss about them in requirement chapter:

- Interoperability
- Peer review process
- Configurable roles in publishing process

2. 3 Modeling the operations of a scholarly publication

After assessing and judging the value of the current systems and due to key role of peer review in Front-End part of knowledge platform a common model for peer review process will be explained. Proposed model does not go to detail of publishing process and try only to explain the common roles and steps in surveyed system.

2.3.1 The main roles and their responsibilities

- Readers: Simple role with limited access. After registration receive a notification email. Reader is permitted to read the article and comment on article.
- Author: The first step is to be taken by the author. It is able to submit manuscripts with supplementary file such as Meta data, etc according to the instructions issued by the journal editor. Track the submission in review and editorial process
- Editor: The main role in the system. It is permitted to supervise all process such as referee and editing. It is able to assign reviewer, copy editor and typography for manuscript. Editor works becomes difficult only when there is significant disagreement in the reviewer's suggestions. In such case the editor may make a final decision based on the own opinion or after consulting additional referees.
- Reviewer (referee): Each journal has an editorial board that includes a number of referees who are responsible for reviewing and evaluation submitted manuscript and supplementary files. Also it is permitted to submit the result of review for editor and put comment for editor and author. Each referee independently advises the editor whether to accept or to reject the paper.

2.3.2 Essential steps in peer review process

1. **Registration**

To have access to published data unregistered user must register to get role in the system. It is first step in publishing process. There are some basic roles in each knowledge publishing system. User can register as a reader, author, referee, etc. Some roles are not accessible for self registration such as Editor. This kind of registration is accessible for administrator only. Some fields in registration method are mandatory.

Email address is momentous field that must be entered correctly because future communication with user is handled by email.

2. **Submission**

Submitting manuscripts is fundamental part of system. Users with author role are able to submit manuscript for publication. There are five steps for submission:

- Check off submission check list and copyright notice.
- Upload submission by providing a choose file window for uploads the file from the computer and renames it following the journal's conventions.
- Enter Meta data such as title, abstract and indexing.
- Upload supplementary files. It is an optional step allows Supplementary files such as source file, figure and table will be uploaded.
- Confirmation the submission's progress. Then manuscript will be viewed through the editorial process.

3. **Editorial board**

   Editorial board decides to reject the submission or begin the process of peer review. Editor can see all articles and review their process status. Article status can be reviewing and editing. Editor role is able to assign an article to reviewing process and notify reviewer by email, sending the article to editor section or prepare for publication. Editor should be able to review unassigned manuscripts, under review article, and preparing the articles for publication. Editor also is able to monitor the emails that reviewer sent for author. Ultimately editor makes a decision and notifies the author.

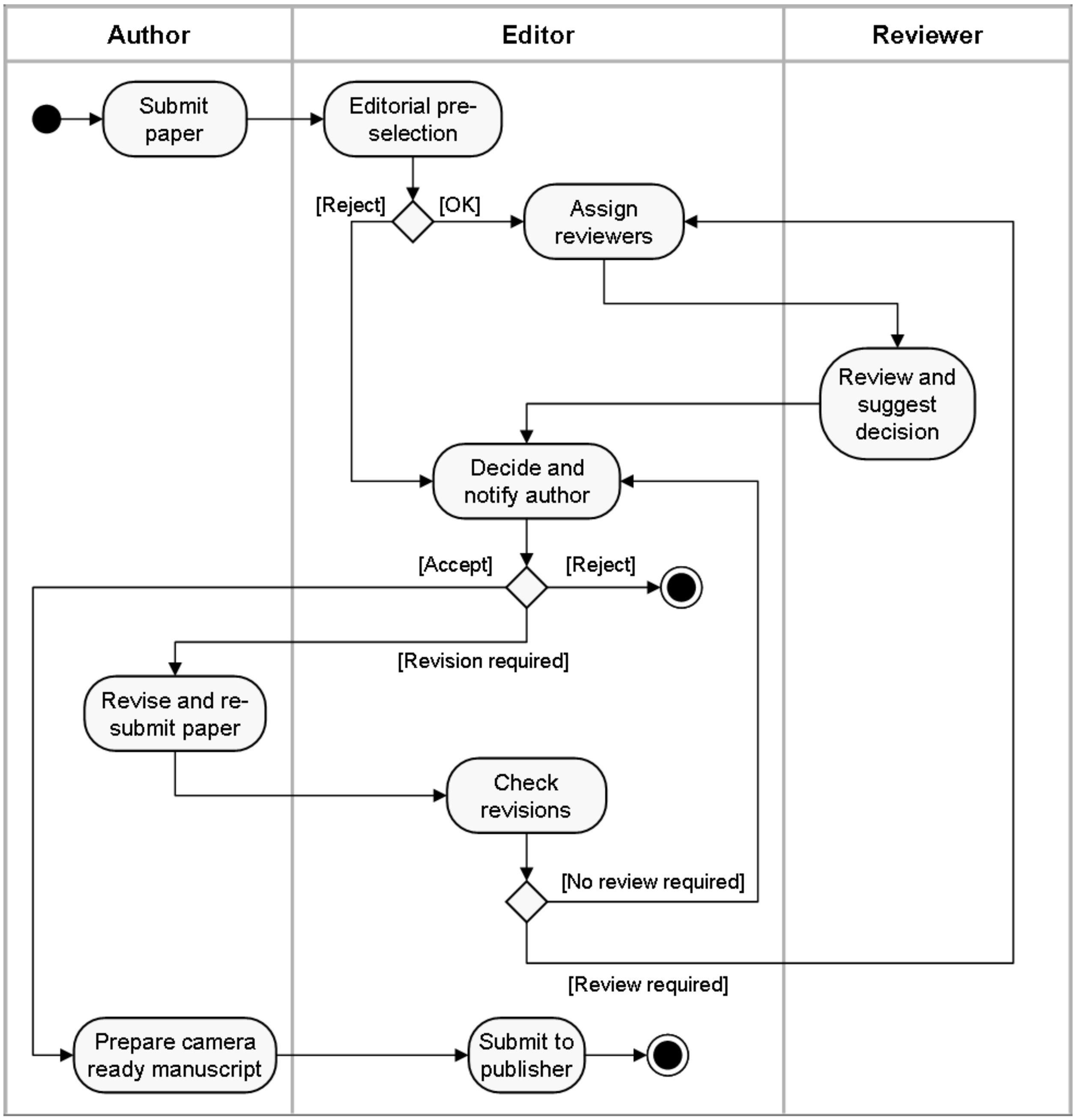
4. **Review the manuscripts (referee)**

   At the first stage reviewer should response to the editor whether he/she accept the review or not. This notification can be done by email. Then if the response be acceptance review can be started. Referee can send email for editor alone or share email for editor and author. It is able to attach more documents and sent them to editor by final recommendation such as accept, reject or revision.

5. **Copy editing**
   Copy editor job is review the submission and improve the clarity, grammar, references and formatting. First copyeditor can edit the author document (for example by using Microsoft word) and send it for author and notify it by email. Then author has opportunity to apply changes on the article and author should response to all suggestion and queries that copyeditor asked. Once the author complete the review editor will be notified by an email. If copyeditor accept the changes submission should be uploaded for editor and notify it by email.
6. **Layout editor**
   It is responsible for converting the word document to PDF or HTML file that is ready to publish. After edit layout it should notify editor by sending an email.

Front-End aims to improve the publicaion steps in following aspect:

- Enhancing speed of process steps such as submission, viewing and feedback of documents among authors, editors and reviewers.
- Reducing costs of review process such as printing cost.
- Allowing for efficient assignment and tracking of reviews, including automatic reminders and enforcement of deadlines.
- Tracking records of actions performed during publish process.
- Archiving data
- Configurable roles



The process of publishing a paper in a journal (8)

2.4 Web 2.0

2.4.1 Overview

Since the inception of the Web in 1993, a considerable amount of the trading and business has moved to Web platforms. For example, Search is a primary activity on the Web today, new models for doing business such as e-commerce and electronic banking (2). The World Wide Web (Web) is a system of interlinked, hypertext documents that run over the Internet. It contains text and multimedia such as images, movies, music, etc. Web pages are written in HTML language. Each page has its own URL (Uniform Resource Locator). Hyperlinks connect pages with each other.

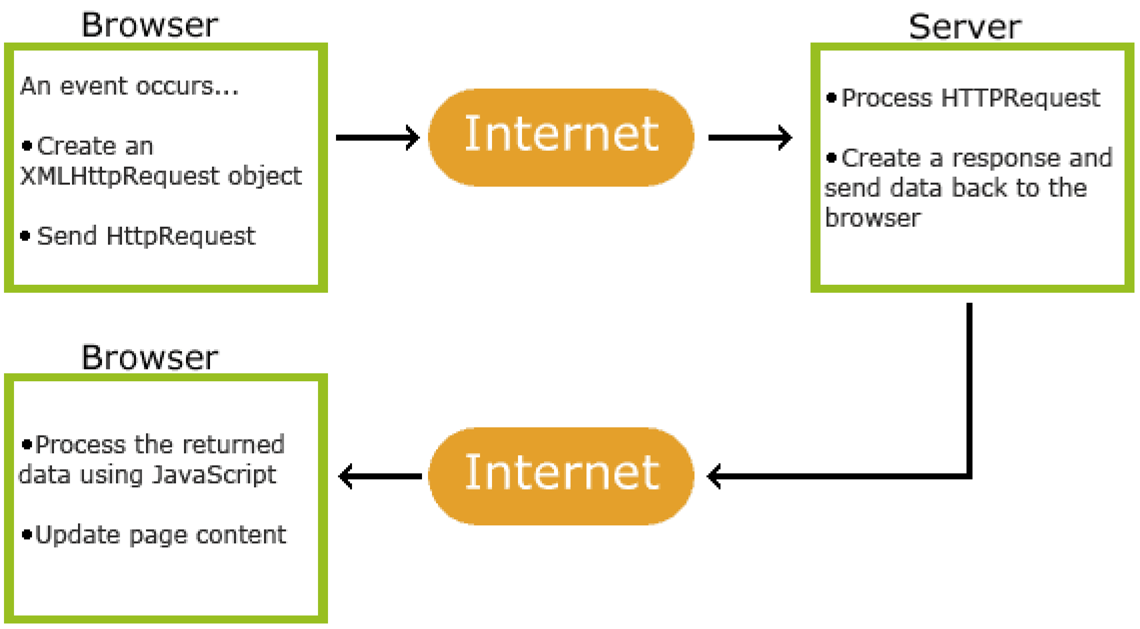
There has been significant progress in hardware and networking technology, and there have also been a number of numerous advances in the software area, which together have helped to make the Web an extremely popular and widely used medium.

The term Web 2.0 was invented in 2003 to represent the new trend which was noticeable in the Web technology. Web 2.0 facilitate community and its effects was more in the way people and businesses using the Web as a collaborative community sites such as blogs, social networking sites, wikis etc. Use of the Web 2.0 term was to distinguish between the many-to-one idea of the Internet as it had been, and the new many-to-many way of thinking that "Web 2.0" came to represent. By using web 2.0 users contribute content and therefore whose content gets richer and more accurate.
Publishing industry power has significant improvements. Web 2.0 causes publishing no longer communicates one-way. Web 2.0 can facilitate the process of searching. Web 2.0 brings easy and secure access to data and desktop functionality for knowledge platform.

2.4.2 Web 2.0 Technologies

2.4.2.1 Ajax

Asynchronous JavaScript and XML (Ajax) is the most technology often used in Web 2.0 applications. It uses JavaScript to upload and download data from the web server and update parts of a web page without reload the pages. This method allows pages to function more like desktop-based applications rather than as old fashioned static content pages. It is a technique for creating fast and dynamic web pages. A good example of Ajax use is Google website. In Google search engine when user type the first word of a sentence search engine will display suggestion information that aids the user to find the required information without reloading the page. Examples of applications using AJAX are Google Maps, Gmail, Youtube, and Facebook tabs. The data format in Ajax request is XML or JSON, two widely used data formats.


Browser
An event occurs...
• Create an XMLHttpRequest object
• Send HttpRequest
Internet
Server
• Process HTTPRequest
• Create a response and send data back to the browser
Internet
Browser
• Process the returned data using JavaScript
• Update page content

How AJAX Works (9)

AJAX is based on existing standards. These standards have been used by developers for several years (9). AJAX applications are independent from browser and platform. The XMLHttpRequest object is used to exchange data with a server and made it possible to update parts of a web page without reloading the whole page. All modern browsers support the XMLHttpRequest object. Asynchronously requests are a huge improvement for web developers. Many of the tasks performed on the server are very time consuming and it could cause the application to hang or stop.

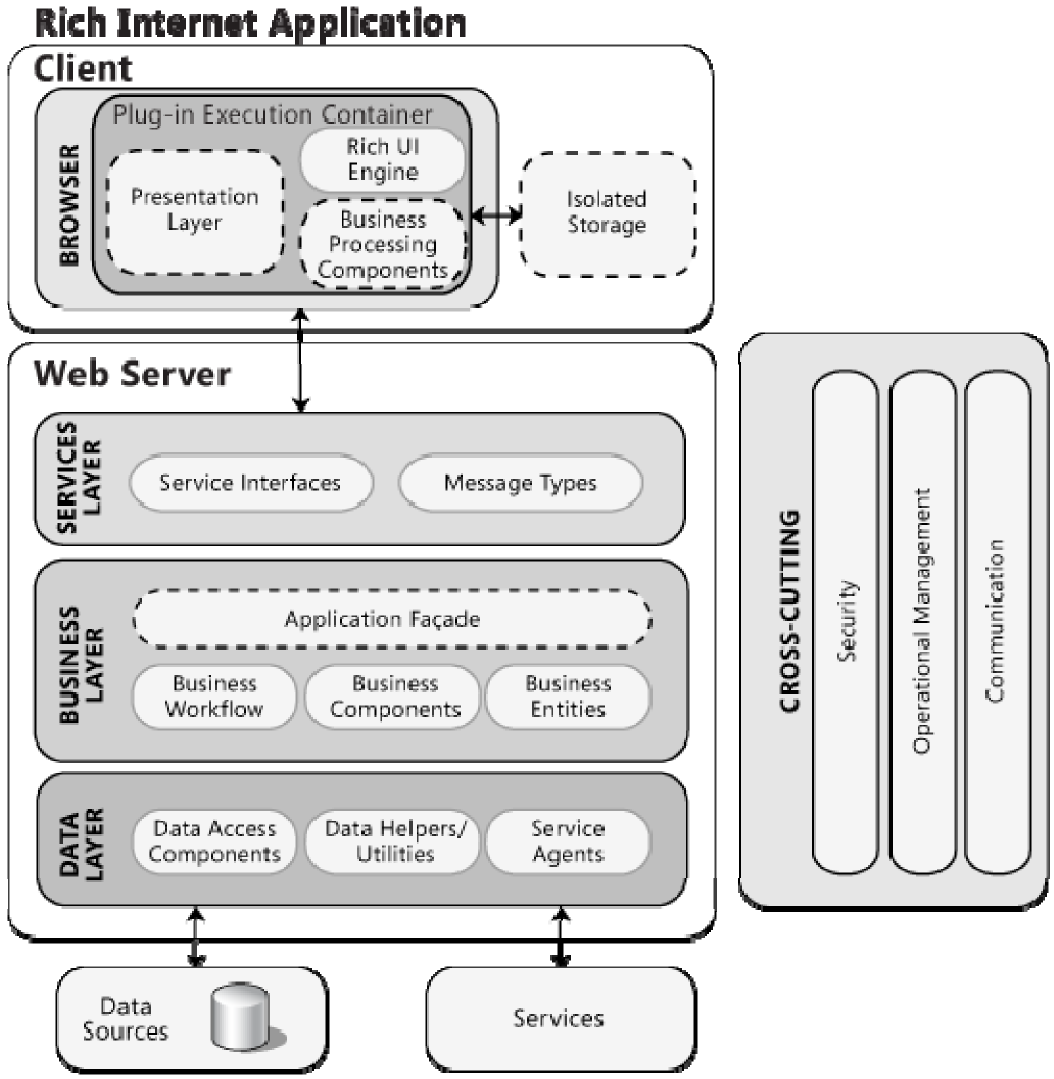
It is possible to perform browser-independent Ajax queries using jQuery (Javascript library). jQuery is free, open source software under the MIT and the GNU General Public License. jQuery is designed to make it easier to navigate DOM elements, handle events, and develop Ajax applications. Using these facilities, developers are able to contribute to the creation of powerful and dynamic web pages.

jQuery advantages:

- Lightweight open source java script library
- Very fast
- Support all browser
- Extensible (provides capabilities for developers to create plugins)
- Nicely handles DOM manipulations

2.4.2.2 Rich Internet Application (RIA)

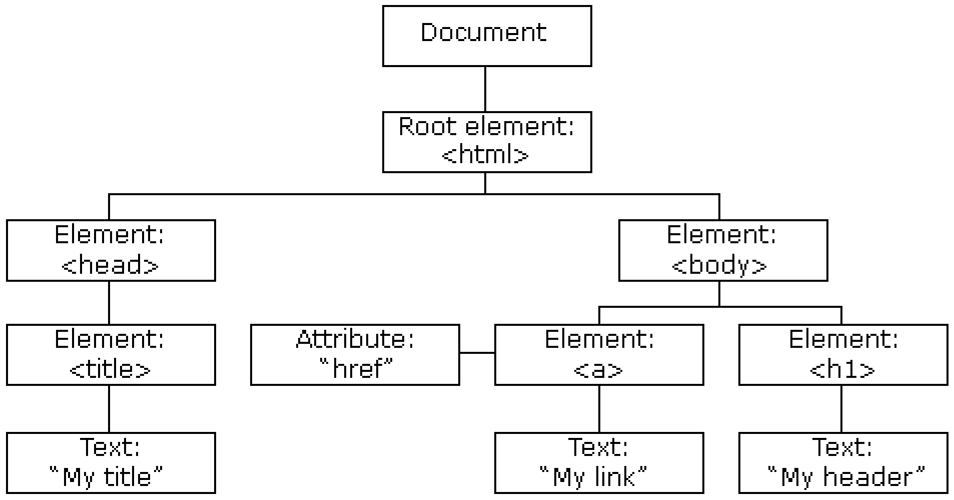
RIA allows the client system to handle local activities and deliver the same features and functions normally associated with desktop applications. For security purposes client portions run within a special isolated area of the client called a sandbox. The sandbox limits access to the file and operating system resource on the client side. RIA provides the end user with an interface that is faster and more responsive than traditional applications. RIA Technologies like Flex, Silverlight, Adobe AIR, and JavaFX are growing in popularity. They provide a richer user experience, mainly through asynchronous communication that lets the user continue interacting with the application while the server is processing requests. JavaFX is a client platform developed to enable easy implementation and deployment of rich Internet applications for desktops, browsers and mobile devices. It is a cross-platform, cross-browser and cross-device technology. It enables to easily integrate audio, video, graphics to program. JavaFX applications can run on any desktop and browser that run the JRE and on top of mobile phones running Java Platform, Micro Edition (Java ME). RIAs support rich graphics and streaming media scenarios on benefits of a Web application. They run inside a browser plug-in, such as Microsoft Silverlight and JavaFX, as opposed to extensions that utilize browser code, such as Asynchronous JavaScript and XML (AJAX). It is a client-side application that handles the presentation in a Web application.



Architecture of a typical RIA implementation. Broken lines indicate optional components. (10)

2.4.2.3 Web Standards

- DOM
  The Document Object Model (DOM) is a cross-platform and language-independent standard for representing objects in HTML, XHTML and XML documents and a way for accessing and manipulating HTML documents. DOM is a W3C (World Wide Web Consortium) standard. It is used to get, change, add, or delete HTML elements.



HTML DOM Tree Example (11)

- XHTML
  It is a combination of HTML and XML (EXtensible Markup Language) that is stricter and cleaner version of HTML. XHTML 1.0 became a W3C Recommendation January 26, 2000.XHTML is properly nested, lowercase and elements must have a closing tag. The DOCTYPE declaration is mandatory in XHTML. DOCTYPE declaration defines the document type. XHTML pages can be validates by WEC validation module.

- CSS

  It stands for Cascading Style Sheets. Styles define how to display HTML elements. Whereas the HTML is the content, the style sheet is the presentation of that document. CSS makes it very easy to change the style of a document. By using CSS all of the style and layout is removed from the html document, so the html file size is smaller. Because CSS file is downloaded just once by browsers and it is re-used for different pages on a web site it causes reduces the bandwidth requirements of server and causes a faster visit for other visitors.

- RDF

  Resource Description Framework (RDF) is a standard for data interchange on the Web. It is an initiative of the World Wide Web Consortium (W3C). RDF has features that facilitate data merging even if the underlying schemas differ. It specifically supports the evolution of schemas over time without requiring all the data consumers to be changed. (12) . RDF is about metadata for Web resources. By resources any object can be found on the Web. Essentially, RDF is a standard for developing tools and applications to use a common syntax for describing Web resources. RDF is a means to express properties of a resource and to associate values with these properties. RDF is syntax independent but can be expressed in XML. Someone believe that using XML as a tool for semantic interoperability will be ineffective in the long run (13).

  RDF is a means to express properties of a resource and to associate values with these properties. However, RDF information can be understood by machine because it keeps data in a formal way. By using RDF resources can be described in a way that software can understand it. Friend-of-a-friend (FOAF) was one of the first applications of RDF that was designed as a Semantic Web version of a personal homepage. It captures metadata about people. The FOAF provides a rich vocabulary to describe personal information such as name, mailbox addresses, homepage URLs, blogs, etc.

The Semantic Web try to build a new World Wide Web architecture that enhances content with formal annotations. Consequently, browsing and searching in the web space is easier. The Resource Description Framework (RDF) is a model for metadata description. Semantic Web brings a significant improvement in data sharing, linking, merging and interoperability which can enrich the structured data already managed by digital libraries and web publication. Interoperability is defined as the ability of digital library components or services to be functionally and logically interchangeable (14). There is a number of popular approaches towards interoperability that are used for interoperability such standardization efforts for Web services like SOAP and WSDL.

2.5 Literature Summery

2.5.1 Problem

Electronic publishing is a new area for dissemination scholarly work which is in an electronic (paperless) form and delivered electronically. Web publishing provide unique benefits such as making scientific research available more quickly, with lower cost and more accessible for people, especially scientists from non-developed countries without limitation by the constraints of the number of print pages .

Peer review process in online publication brings significant benefits such as fewer errors, less duplication and control the quality in disseminating scientific information. For quality control and accuracy, research works must exercise peer review or editorial quality control to be included before publication.

In addition, peer review is often a slow process. After a manuscript is submitted to a journal for consideration, there can be a delay from several weeks to several months before it is published in a journal makes journals a less than ideal tools for disseminating research.

2.5.2 Solution

Web 2.0 brings interactive pages and user participation such as social networking. It offers fundamental changes in communication technologies. In the same way Web 2.0 allows prepublication review process of academic papers can be done online by using more interactive technologies.

We are going to use new technologies to improve the online publication process in following area:

- ✓ Online submission
- ✓ Support parts of an electronic referring process
- ✓ Web base reading tools for content
- ✓ Time reminders for deadlines of reviews and email notification
- ✓ Interoperability to access to material through interoperable repositories
- ✓ Online tracking manuscripts status
- ✓ Control permission access to article for individuals or groups of users
- ✓ User-Friendly interface and desktop functionality in different stages
- ✓ Cross referencing
- ✓ Semantic ranking

Web 2.0 and RIA technologies can be used in Front-End to enhance the mentioned features in the platform. These technologies can facilitate the process of knowledge publishing by provide a secure access to core knowledge database and desktop functionality for above mentioned features.

Chapter 3

Design & Implementation

During this chapter design and implementation technique are discussed. Patterns, framework and the tools which were used for modeling the application layers in the project are listed and discussed.

In Section 3.1 it is presented why Java EE was selected as a platform for server programming.

In section 3.2 the application pattern is explained. It discussed how MVC design pattern brings about better organization and code reuse in Knowledge Platform.

In section 3.3 the framework that is used for developing the Front-End is discussed. It shows how framework facilitates the development process and reduces the overall development time and how it helps to reduce application complexity.

Section 3.4 shows how selected tools and framework address requirements that were mentioned in chapter 3 such as extensibility, Interoperability, Security, Semantic ranking and Peer review.

3.1 JEE Platform

Java Platform, Enterprise Edition (Java EE) is the industry standard for enterprise Java programming (15). Over the years, the Java EE platform has grown and matured and it is able to cover a wide range of requirements in during development process of enterprise web application. It has active community and marketplace for additional activity such as frameworks, Libraries, and enterprise web applications that work with the platform. It provides developers a powerful set of APIs such as JDBC, RMI, e-mail, JMS, web services, XML while reducing development time, reducing application complexity, and improving application performance.

3.1.1 Why JEE

With using JEE developers are able to design and implement the business logic into components according to business requirement. It provides functionality to deploy distributed and multi-tier application, what are based on modular components on the application server. There are a wide range of Java development tools and open source IDE.

JEE support multi-tiered application by installing application on different machines depending on the tiers. Tiers can be categorized in client tier, web tier, business tier, data model tier. Two main JEE features are *compatibility* and *distribution*. Compatibility means that JEE applications are compatible with different platforms and can easily be deployed different platforms. We also are able to compress all components of an application such as servlet, pages, POJO classes into a single archive file that easily can be deployed and distributed across platforms and web servers (16).

Java EE has separated the different roles of developing an application. Application assembler is a person who assembles a set of J2EE modules into a packaged as an Enterprise Archive such as Web Archive (WAR file). Deployer takes the packaged application and put the Enterprise Archive on the server. It configures the environment for example user management for the application, mapping the security roles, correct properties, XML configuration files and manifest for the web application. Web component developer specializing in the application of JavaServer Pages and servlet technologies used to provide Web services and dynamic Web content.

3.1.2 Enterprise technologies

The J2EE platform includes a number of standard APIs for accessing existing information systems. For instance The JDBC API is used for accessing relational data from the Java programming language (15). The Java Transaction API (JTA) is for managing transactions over heterogeneous enterprise information systems.

The JavaMail API is used for sending and receiving e-mail. Java APIs for XML provide support for implementing Web services in the J2EE platform.

J2EE provide a mechanism for supporting distributed applications, with no application development effort. Because JEE application can be run on multiple systems, Web containers can automatically balance load in response to various demand. The J2EE provides flexible and secure control model for accessing to application services. Developers can specify the security requirements at the method level to ensure that only authorized users can access specific service and data. Both Enterprise JavaBeans technology and Java Servlet APIs provide programmatic role-based security mechanism that can be used during deployment time.

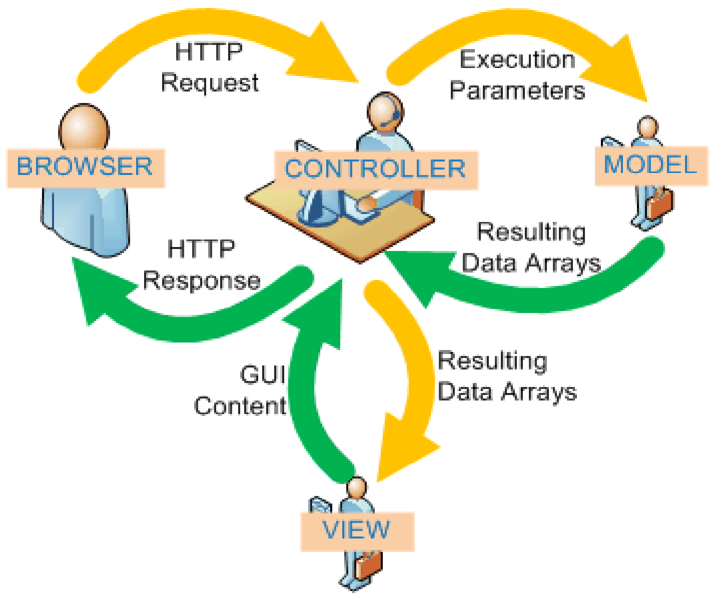
3.2 Design Pattern

A pattern aims to propose a solution for particular problems so it proposes a set of values to guide the designer toward a decision that is best for their particular application.

Model-View-Controller (MVC) is architectural design pattern for enterprise applications. MVC was first described in 1979 by Trygve Reenskaug on Smalltalk. MVC provide a powerful pattern to develop dynamic web applications that support a clean separation of concerns. MVC aims to decouple models and views to reduce the complexity in designing application's layers to increase flexibility and maintainability of source code. Most web application frameworks use MVC design pattern. MVC organizes a layered application into three separate modules. Model is used for data representation and business logic. View provides data presentation and user input. Controller is responsible to dispatch requests and control flow.

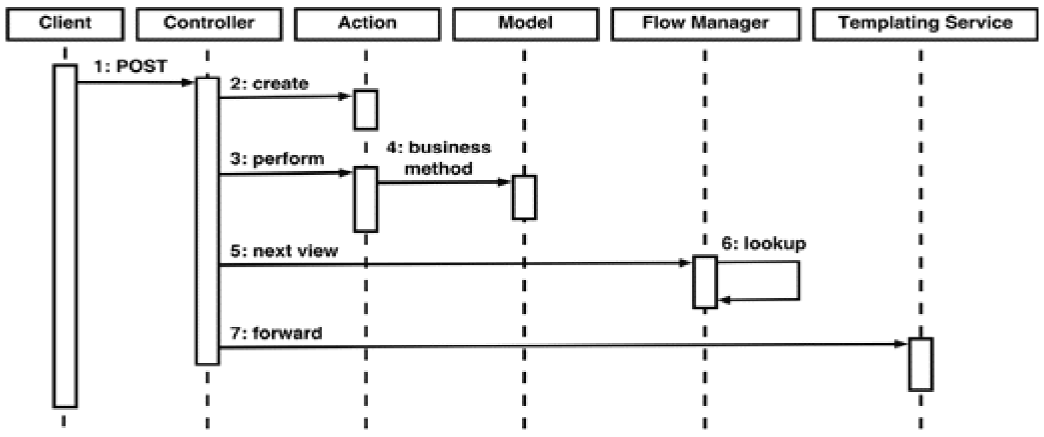
3.2.1 Workflow

Front Controller centralizes an application's request processing and view selection in a single component. The Front Controller receives requests from the client and dispatches them to the application models. In the J2EE platform, a Front Controller is typically implemented as a servlet. The sample application's Front Controller servlet handles all HTTP requests and it plays a separator role between logical part (models) and view engine (GUI generator).



Front Controller handles HTTP requests

Controller maps incoming requests to appropriate model. After getting feedback from the model it is sent to view engine to generate the final response. Usually, the response has html format.



MVC Workflow (15)

3.2.2 MVC Benefits

Placing business logic and presentation code in separate layers has valuable benefits. The business layer provides only application functionality, with no relation with presentation. The presentation layer presents the data and input prompts to the user as web pages (View) and business layer is responsible for application functionality. This separation brings several important benefits (15):

- ✓ Minimize the impact of changing the code inside a layer on the other layers.
- ✓ Increases maintainability by keeping business logic in a separate component and accessed referentially. Business logic can be modified in one place and causes behavior changes everywhere the component is used.
- ✓ Business logic can be used by multiple client types because logic is provided for client independently and other view technologies such as desktop application can use the business objects easily.
- ✓ Separating developer roles is possible by layering. Developer may specialize in only one layer (front-end developers usually are involved in presentation layer but web component developers are involved in business layer). Separating business logic and presentation allows developers to concentrate on their area of expertise.

3.3 Framework

A Framework is a package of software libraries. They improve the efficiency, productivity, quality, reliability by providing a set of libraries and defining the flow of control for an application. In frameworks unlike libraries flow of control is dictated by the framework not by caller (inversion of control). Framework has a default behavior. Framework can be extended by the user but is not allowed to be modified.

A web application framework is a type of framework that helps developers specifically to design and develop web applications. Frameworks provide basic common functionality to the web applications such as session management, data persistence and data mapping, template systems, caching, security, URL mapping and web service. By using an appropriate framework, a developer can often save a significant amount of time building a web application.

3.3.1 Frameworks Features

- Web template system: Templating frameworks are built to simplify the development of web application's user interfaces. Template allows defining page fragments which can be assembled into a complete page at runtime. In order to reduce the duplication of common page elements or embedded within other tiles to develop a series of reusable templates.
- Caching: Web caching is the caching of web documents in order to reduce bandwidth usage and server load. It results in users receiving content faster and can drastically improve the user experience. Caching can be handled in different level such as HTTP response header and database. Http response caching can be handled in web browsers and some Internet networking components, such as proxies. Database cache can use Object-relational mapping (ORM). It can load related business objects on demand. To keep it simple, this usually involves many small queries to load data as needed.
- Security: Some parts of web applications often require that be secured. Standard security mechanisms can satisfy some security requirements but in some cases we need to use a customized approach. Some web application frameworks support authentication and authorization. In these cases framework enables the web server to identify the users of the application, and restrict access to functions based on some defined criteria. Some Web application framework provides advanced authentication, authorization and other security features for enterprise applications such as such as SAP, Oracle EBS, PeopleSoft, and Siebel.
- Database access and mapping: Many web application frameworks provide API for connecting to database. They support working with a variety of databases with no code changes and work with higher-level concepts. Provide Object Relational Mapping (ORM) which will map objects to

tables in database is a programming technique for converting data between incompatible type systems in object-oriented programming languages. Some frameworks support ORM technique to reduce the amount of code that needs to be written.

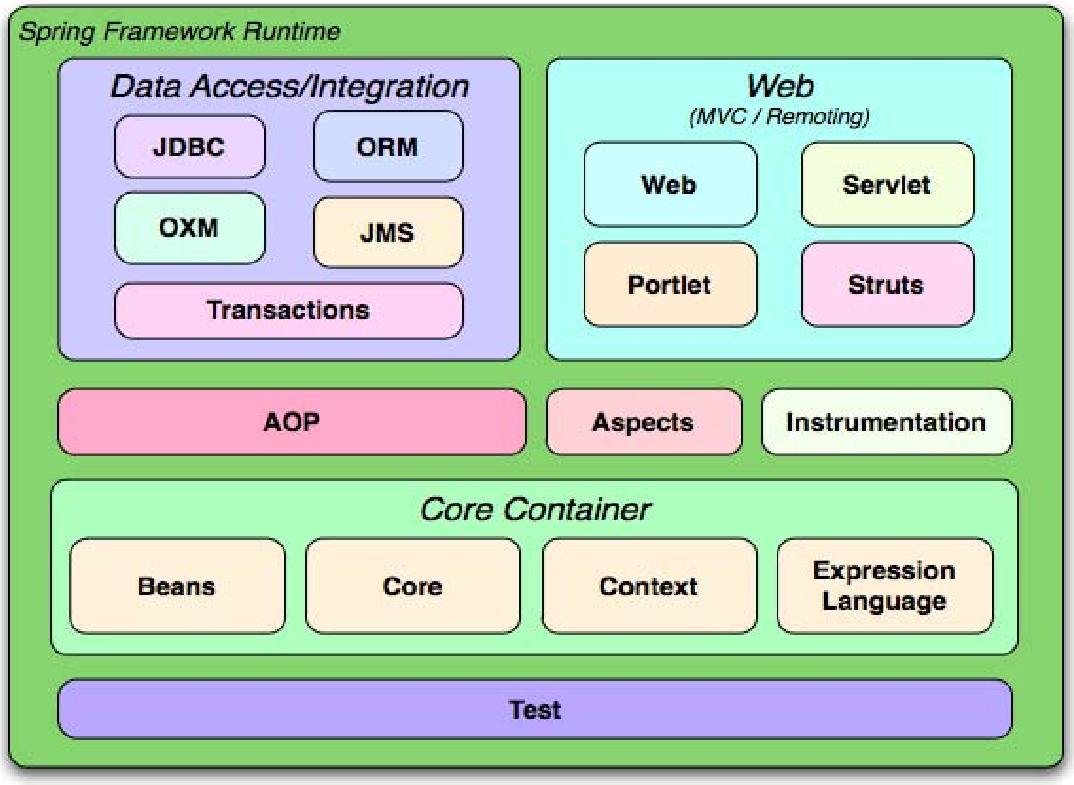
- URL mapping: The URL mapping technique helps you to map a specified URL with another URL. As administrators create the URLs, a URL mapping system that uses pattern matching or URL rewriting allows more friendly URLs to be used. It brings other benefits such as increasing the simplicity of the URLs and it also allows search engines to have a better indexing. Another reason for translating URLs is usability. URLs with query strings cannot be memorized and they are not user friendly. Users prefer shorter URLs with meaningful keywords. One more reason for URL rewriting could be if application's directory structure changes it is possible that we don't lose incoming links from other sites because everything will stay same for outside world, but our application will work differently. Also, there is some security risk if we expose a query string variable to visitors.

3.3.2 Why Spring Framework

The Spring Framework is a lightweight solution for building enterprise applications. Spring Framework is a Java platform that provides comprehensive infrastructure support for developing Java applications. Spring is modular and we can select those parts that are needed without having to bring in the rest. It offers a full featured MVC framework and enables us to integrate Aspect Oriented Programming (AOP) transparently into our software.

3.3.2.1 Spring modules

The Spring Framework consists of features that are organized into about 20 modules (17). These modules are grouped into Core Container, Data Access, Web, Aspect Oriented Programming (AOP), Instrumentation, and Test, as shown in the following diagram.



Modules diagram (17)

1) Core Container: The Core Container consists of the Core, Beans, Context, and Expression Language modules. The Core and Beans modules provide the IoC and Dependency Injection features. The Context module support internationalization and basic remoting. The Expression Language module provides a powerful expression language for setting and getting property values, property assignment, method invocation, accessing the context of arrays, collections and indexers, logical and arithmetic operators, named variables, and retrieval of objects by name from Spring's IoC container.
2) Data Access/Integration: This layer consists of the JDBC, ORM, OXM, JMS and Transaction modules. The JDBC module provides a layer that removes the need to do tedious JDBC coding. The ORM module provides object-relational mapping APIs such as including Hibernate provides simple declarative transaction management. The OXM module supports Object/XML mapping implementations. The Java Messaging Service

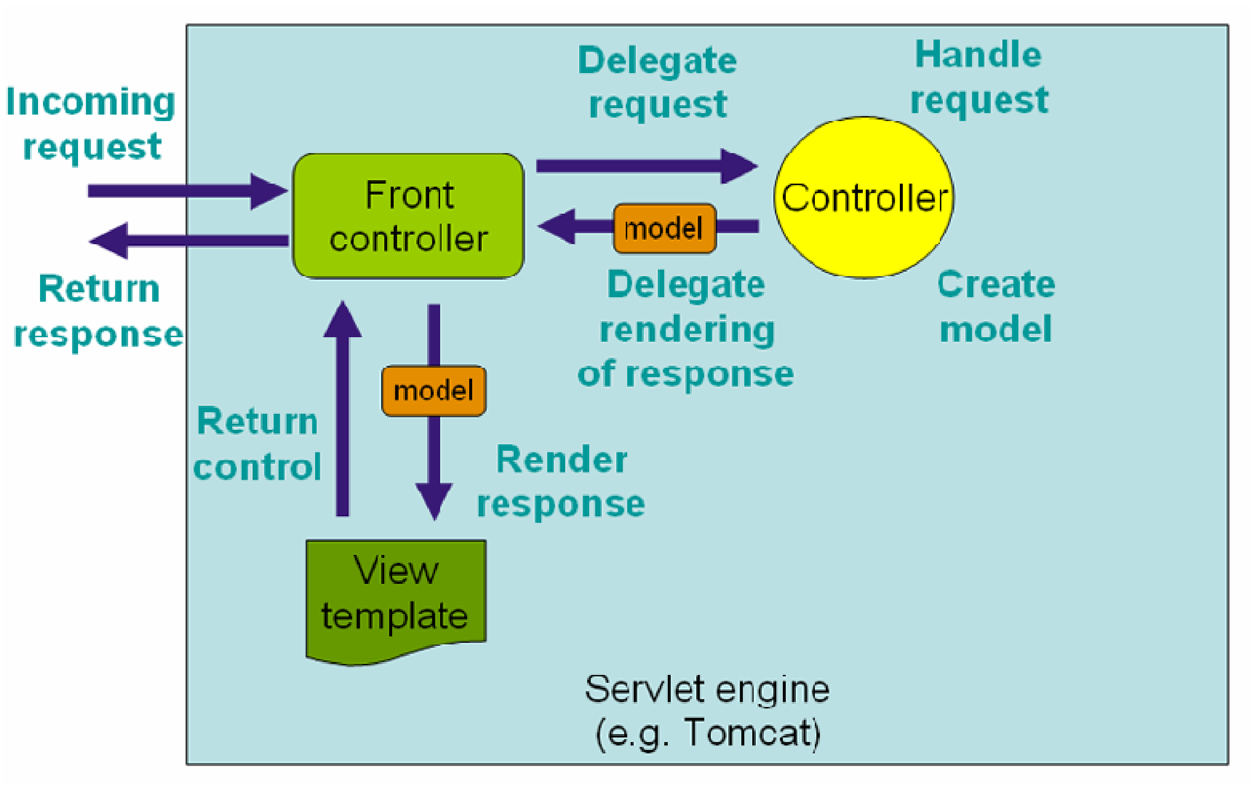
(JMS) module contains features for producing and consuming messages. The Transaction module supports programmatic and declarative transaction management.

3) Web layer: Consists of the Web, Web-Servlet, Web-Struts, and Web-Portlet modules. It provides basic web oriented integration features such as multipart file-upload functionality and the initialization using servlet and application context. The Web-Servlet module contains Spring's model-view-controller (MVC) implementation for web applications.
4) AOP: Aspect oriented programming provides method-interceptors and pointcuts to cleanly decouple code that implements functionality that should be separated.
5) Test: This module supports JUnit. It also provides mock objects that you can use to test your code in isolation.

Spring's Role in the Front-End

After discussing about MVC pattern and surveying Web application frameworks and discovering the Spring modules, now it is turn to define role of the Spring framework in Front-End development process. During developing process of Front-End we are going to use Spring Web model-view-controller (MVC) framework. Features and components of Spring that we will use to develop Front-End and benefits that it can bring for us is discussed in following categories:

- Front Controller: Spring Web MVC is request-driven, designed around a central servlet that dispatches requests to controllers and offers other functionality that facilitates the development of web applications. The request processing workflow of the DispatcherServlet is illustrated in the following diagram.



Front-End Use DispatcherServlet as a Front Controller (17)

- Mapping URL: Spring brings URL mapping advantages to Front-End. URL mapping technique in Front-End increasing the simplicity of the site and allowing for better indexing by search engines. Users prefer shorter URLs with meaningful keywords. Also with URL mapping changing application's directory structure can be done without losing incoming links from other sites. Spring supports URI Templates to access parts of a request URL in handling methods. Controller defines URL Mappings, which associate URLs to Operations and Views. URL Mappings is used to specify which Operations are going to be accessible to web clients. The URL mapping also specifies the template view that will render the response to the request. Every request will have a response, and the URL mapping specifies both the operation that handles the request and the template page that will render the response.

- Spring supports different view technology such as JSP, PDF, Excel File. Supporting multi view technologies is an important requirement for Front-End. Spring supports multiple view resolvers. We can chain resolvers to override specific views in certain circumstances. For example we can have two views resolver. The general one supports dynamic JSP file and the specific view resolver for Excel views.
- Internationalization: Spring's architecture support internationalization and localization. Supporting multiple locales means when a request comes in, the DispatcherServlet looks for a locale resolver, and if it finds one it tries to use it to set the locale. For example ResourceBundleViewResolver supports view resolvers with regard to internationalization. It can return a different View implementation for the same logical view name, based on the user's Locale. Also Spring provides powerful tag library to renders a web page's message and raise error from an external message properties file to the output. Since web applications such as knowledge platform are accessible anywhere that Internet service is provided, then supporting multi languages feature in Front-End can be very useful facility.
- Spring's multipart (fileupload) support: This feature is important for Front-End because uploading file is one of the main part of peer review process. When the Spring DispatcherServlet detects a multi-part request first create a form with a file input that will allow the user to upload a form. The encoding attribute (enctype="multipart/form-data") lets the browser know how to encode the form as multipart request. The next step is to create a controller that handles the file upload.
- Handling exceptions: Spring ease the pain of unexpected exceptions that occur while your request is handled by a controller that matched the request and provides a more flexible way to handle exceptions.
- Form tag library: Spring provides a comprehensive set of data binding-aware tags for handling form elements when using JSP and Spring Web MVC.
- Integrate Struts Tiles: Almost web application should use page template. Front-End also is not an exception to this requirement. Spring supports

integration with Struts tiles. Tile is a templating system. It can be used to create a common look and feel for a web application. Tiles help to simplify the development of web application user interfaces. It allows defining page fragments which can be assembled into a complete page at runtime. In order to reduce the duplication of common page elements.

3.4 Implementing Front-End

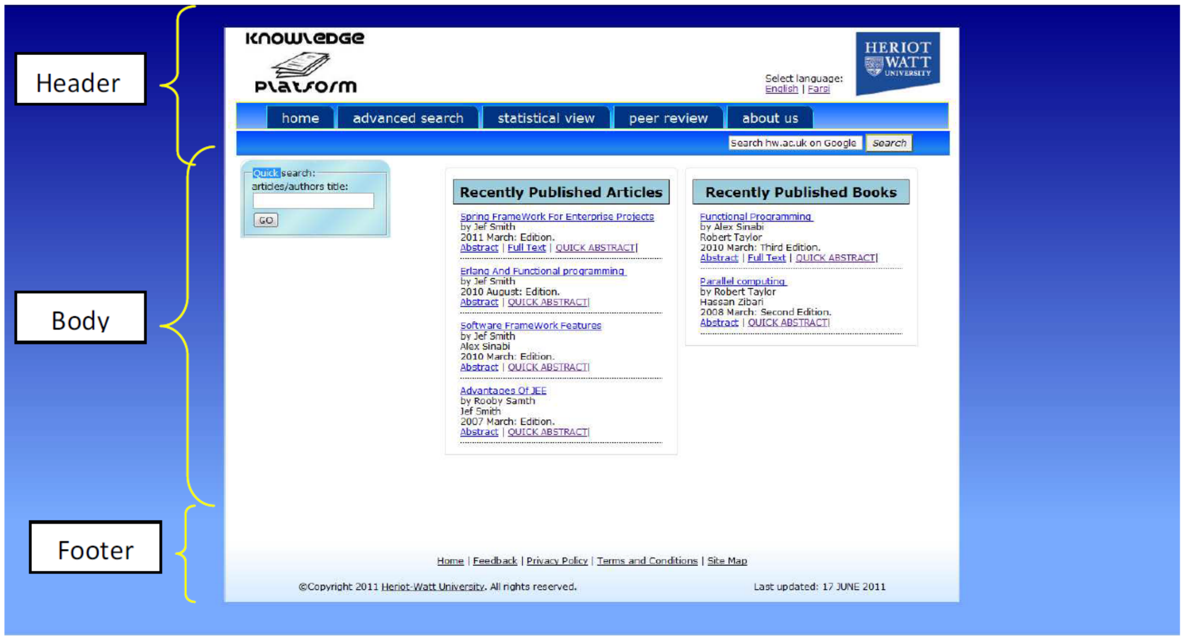
In this section the implementation of Front-End in the Java Enterprise Edition (J2EE) platform is explained in detail. It is based on the concepts introduced in the last sections. In this section we are going to show how we use selected platform and framework to address Front-End's requirements.

3.4.1 Tiles Template

Apache Struts Tiles framework is a layout framework, which allows us to maintain a standard look of header, footer and general theme all of the web pages efficiently. At run time the pagelets are stitched together to generate the final HTML. In Front-End we use a template that contains three elements: a title, a header, and a body displayed vertically. Tiles are defined in an XML configuration file (tiles-definition.xml). The file is loaded at servlet startup that ensures all definitions are loaded at application startup. The layout.jsp file contains the template code. This file does not change.

```
<tiles-definitions>
   <definition name="layout" template="/WEB-INF/pages/jsp/layout.jsp">
         <put-attribute name="title" value=""/>
         <put-attribute name="content" value=""/>
   </definition>
   <definition name="home" extends="layout">
         <put-attribute name="title" value="Knowledge Platform's Home Page"/>
         <put-attribute name="content" value="/WEB-INF/pages/jsp/home.jsp"/>
   </definition>
</tiles-definitions>
```

An overview of the Tiles XML configuration mechanism



Select language:
English | Farsi

Search hw.ac.uk on Google

Quick search:
articles/authors title:

Recently Published Articles

Spring FrameWork For Enterprise Projects
by Jef Smith
2011 March: Edition.
Abstract | Full Text | QUICK ABSTRACT|

Erlang And Functional programming
by Jef Smith
2010 August: Edition.
Abstract | QUICK ABSTRACT|

Software FrameWork Features
by Jef Smith
Alex Sinabi
2010 March: Edition.
Abstract | QUICK ABSTRACT|

Advantages Of JEE
by Rooby Samth
Jef Smith
2007 March: Edition.
Abstract | QUICK ABSTRACT|

Recently Published Books

Functional Programming
by Alex Sinabi
Robert Taylor
2010 March: Third Edition.
Abstract | Full Text | QUICK ABSTRACT|

Parallel computing
by Robert Taylor
Hassan Zibari
2008 March: Second Edition.
Abstract | QUICK ABSTRACT|

Home | Feedback | Privacy Policy | Terms and Conditions | Site Map

©Copyright 2011 Heriot-Watt University. All rights reserved.

Last updated: 17 JUNE 2011

To apply changes into the body, it is just needed to update the "tiles-definition.xml" file.

3.4.2 Internationalization and Localization

The Internet has no boundaries and neither should Front-End. Knowledge platform's user all over the world has different languages. It is an important factor to make Front-End user friendly in different countries using different languages. Internationalization (I18n) is the process of enabling our application to cater to users from different countries and supporting different languages without engineering changes. Localization (L10n) on the other hand, is the process of customizing our application to support a specific location without engineering changes (18).

Advantages of Internationalization and Localization:

- It causes an increase in the number of users because more people are able to use the application.

- Though the cost of localization is high (Each individual page will have to be translated into several languages) but the benefits from sales almost always outweigh the costs.
- It increases the user level of comfort with the application.

Front-End support two languages: English and Persian. Depending on the locale setting of user's browser, the appropriate language will be selected. Also user will be able to select the language from top-right corner of the application. In Spring MVC application "LocaleResolver" interface supports the internationalization or multiple languages features. It displays the message from properties file, and change the locale based on the selected language link. Two properties files to store English and Persian messages. To make Spring MVC application supports the internationalization we register two beans. Firstly, CookieLocaleResolver is used to inspect a Cookie that might exist on the client to see if a locale is specified. If so, it uses the specified locale. By using its properties we specify the name of the cookie as well as the maximum age.

```
<bean id="localeResolver" class="org.springframework.web.servlet.i18n.CookieLocaleResolver">

    <property name="defaultLocale" value="en" />

    <property name="cookieName" value="clientlanguage"/>

    <!-- in seconds. If set to -1, the cookie is not persisted (deleted when browser shuts down) -->

    <property name="cookieMaxAge" value="-1"/>

</bean>
```

Secondly, LocaleChangeInterceptor is used to enable changing of locales by user. It will detect a parameter in the request and change the locale. Because some languages such as Persian or Arabic have different theme (For example they are right-to-left) I need to add ThemeChangeInterceptor to enable Front-End's users to change language and theme simultaneously.

```
<!-- Declare the Interceptor -->
<mvc:interceptors>
   <!-- Locale Change Interceptor and Resolver definition -->
  <bean id="localeChangeInterceptor"class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor">

        <property name="paramName" value="lang" />
   </bean>
   <!-- Theme Change Interceptor and Resolver definition -->
   <bean id="themeChangeInterceptor"
        class="org.springframework.web.servlet.theme.ThemeChangeInterceptor">
        <property name="paramName" value="theme" />
   </bean>
</mvc:interceptors>
```

Theme interceptor's configuration file

3.4.3 URL Mapping

In Front-End request are divided in two models: First request for resource such as image and CSS files. These kinds of requests are not delivered to DispatcherServlet and directly are handled by web server. The second group of requests will be handled by the DispatcherServlet. The DispatcherServlet is an actual Servlet and it is declared in the web.xml of Front-End. After receiving a request, The DispatcherServlet finds the appropriate Controller with the help of HandlerMapping and then invokes associated Controller.

```
<!-- resources exclusions from servlet mapping -->
<mvc:resources mapping="/style/**" location="/style/" />
<mvc:resources mapping="/js/**" location="/js/" />
<mvc:resources mapping="/pdf/**" location="/pdf/" />
```

3.4.4 Restful Web Service

As project's requirements specified, Front-End should get information from Core (An application that is responsible to handle data layer tasks). Front-End that needs data from the Core use the restful web service to connect to the Core. REpresentational State Transfer (REST) is an architectural style for distributed systems over the WEB. In REST architecture resources are identified by universal resource identifiers (URIs).

Spring framework provides a good way to use rest web service. The RestTemplate is the central Spring class for client-side HTTP access. HttpMessageConverters is responsible for converting the Java objects to HTTP requests and from HTTP responses. REST configuration file is located as following:

```
<!-- Restful Web Service definition  -->
<bean id="restTemplate" class="org.springframework.web.client.RestTemplate">
    <property name="messageConverters">
        <list>
        <bean id="stringHttpMessageConverter"  class="org.springframework.http.converter.StringHttpMessageConverter">
                <property name="supportedMediaTypes" value="text/html"/>
            </bean>
         <bean id="messageConverter" class="org.springframework.http.converter.xml.MarshallingHttpMessageConverter">
                <constructor-arg ref="webServiceXMLConvertor" />
                <property name="supportedMediaTypes" value="application/xml"/>
             </bean>
        </list>
    </property>
 </bean>
```

3.4.5 Validation

An important aspect of creating Web pages for Front-End is validation. Page validation is the process of testing to ensure that end users enter necessary and properly formatted information into web forms. Web pages should be to be able to check that the information users enter is valid. Front-End provides a set of validation controls that provide a powerful way to check for errors and, if necessary, display messages to the user. There are several methods of performing form validation and error recovery that Front-End use them.

3.4.5.1 client-side

Validation and error recovery mechanisms are performed within browser by using a client scripting language such as Javascript. One of the advantages of client-side validation and error recovery is that the user gets quick feedback, and doesn't have to wait for a server round-trip. Client-side validation is widely used, but is not security relevant. An attacker can simply disable JavaScript to bypass the client side validation. Front-End have used JavaScript functions to validate web pages in client side. The sample source code is listed as follow:

```
// Numeric only control handler
jQuery.fn.ForceNumericOnly =function()
{
   return this.each(function()
   {
      $(this).keydown(function(e)
      {
         var key = e.charCode || e.keyCode || 0;
         // allow backspace, tab, delete, arrows, numbers and keypad numbers ONLY
         return (
            key == 8 ||  key == 9 || key == 46 || (key >= 37 && key <= 40) || (key >= 48 && key <= 57) ||
            (key >= 96 && key <= 105));
      });
   });
};
```

Using JQuery for client side validating

3.4.5.2 server-side

Server-side validation mechanisms cannot be easily bypassed or modified. Front-End use following mechanism for server-side validation:

1) Validation using Spring's Validator interface

Spring features a Validator interface that can be use to validate objects. validators report validation failures to the Errors object. The sample source code is listed as follow:

```
@Component
public class NameValidator implements Validator {
    @Override
    public boolean supports(Class clazz) {
        return Name.class.isAssignableFrom(clazz);
    }
    @Override
    public void validate(Object obj, Errors e) {
        Name name = (Name) obj;
        if (name.getFirstName().trim().length()<3)
            e.rejectValue("firstName", "name.firstName");
        if (name.getLastName().trim().length()<3)
            e.rejectValue("lastName", "name.lastName");
    }
}
```

Validation using Spring's Validator interface

Outputting messages corresponding to validation errors is important. As it is clear we used error code and what error codes it registers is determined by the MessageCodesResolver that is used. Front-End have used ReloadableResourceBundleMessageSource class find corresponding message text.

2) Hibernate Validator

Hibernate Validator is useful because it supports declarative validation via Java 5 annotations. By using Hibernate Validator in Front-End we only attach validation annotations to the bean properties and that will define the validation constraints for the bean. Moreover Hibernate Validator isn't tied to the web tier and it is possible to validate beans from within service layers. The sample source code is listed as follow:

```
public class Name {
    @NotBlank
    @Length(max = 40,min=3)
    private String firstName;
    private String middleName;
    @NotBlank
    @Length(max = 40,min=3)
    private String lastName;
```

3.4.6 FileUpload

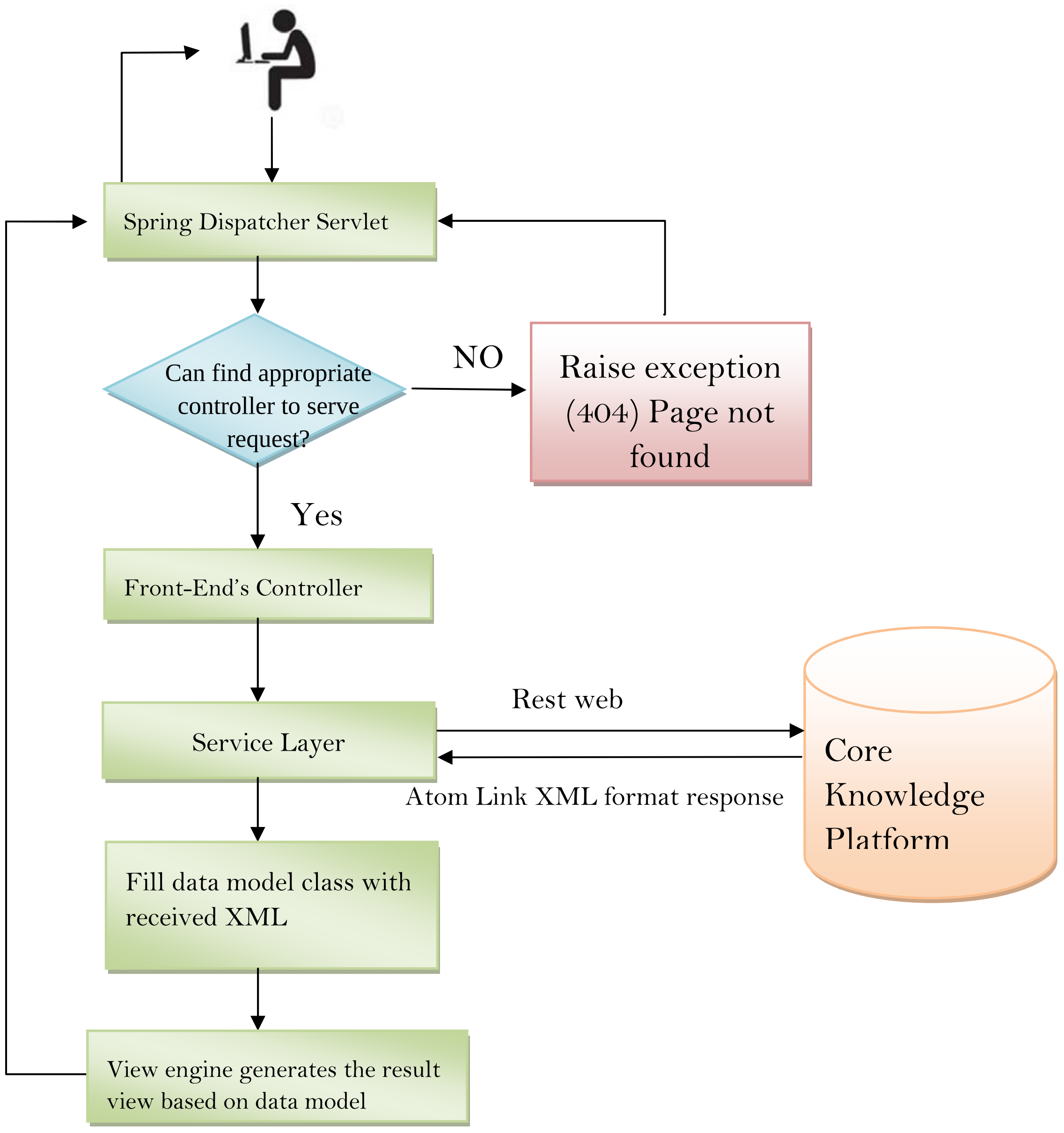
FileUpload can be used in a number of different ways, depending upon the requirements of your application. Front-End choose Apache's commons-upload mechanism. Spring handle file upload in web application with MultipartResolver which use the Apache commons upload library to handle the file upload. Upload file's configuration file is listed below:

```
<bean id="multipartResolver"
class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
   <!-- The maximum file size in bytes -->
   <property name="maxUploadSize" value="10000000"/>
</bean>
```

As it is clear in sample configuration, maximum file size is configurable. FileUpload parses POST HTTP requests with a content type of "multipart/form-data". Front-End keeps all uploaded files in *uploadedfiles* folder at web application folder on the server.

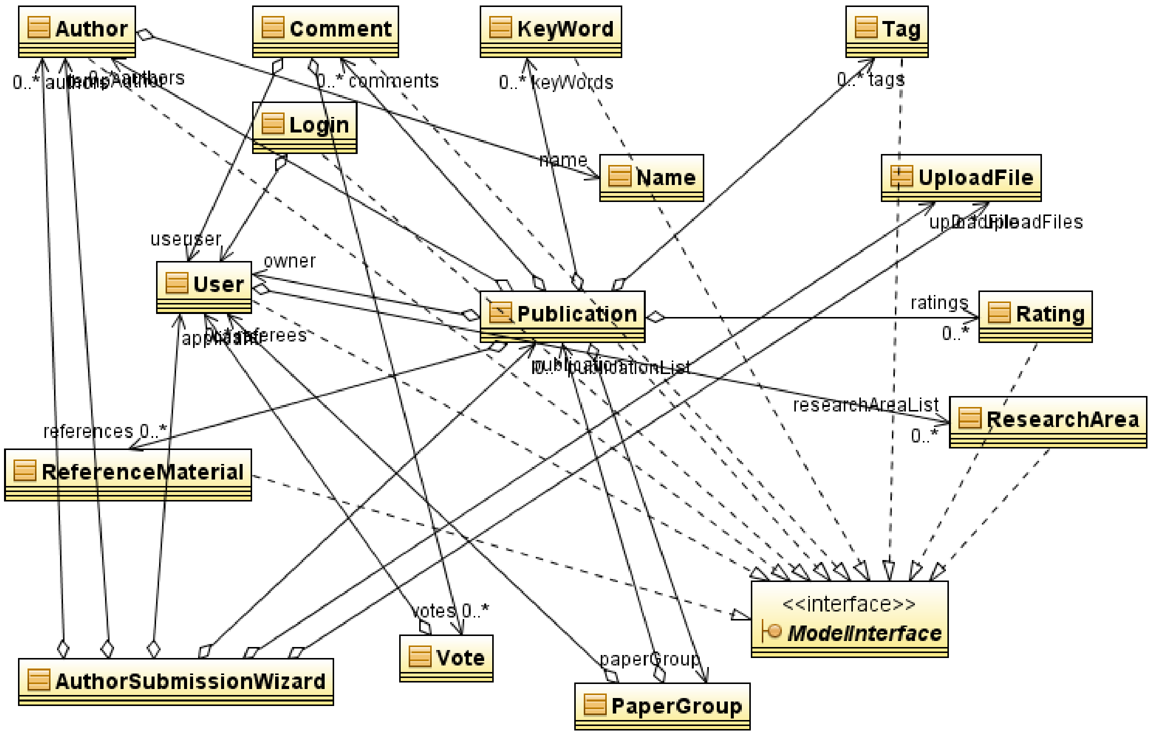
3.4.7 Overall View of Front-End

After describing the technologies and methodologies that are used in the constructing and developing of Front-End, it is turn to describe an overall picture of the Front-End. The request processing workflow of the Front-End is illustrated in the following diagram.


Spring Dispatcher Servlet
Can find appropriate controller to serve request?
NO
Raise exception (404) Page not found
Yes
Front-End's Controller
Rest web
Service Layer
Atom Link XML format response
Core Knowledge Platform
Fill data model class with received XML
View engine generates the result view based on data model

The requesting processing workflow in Front-End (high

Front-End keeps the fetched data in java beans classes. Front-End's data layer classes are illustrated in the following diagram:



As it is clear from the diagram there is a JavaBeans class for each entity that is defined in the data model. JavaBean allows access to properties using getter and setter methods.

More details and the source code can be found in the appendix A.

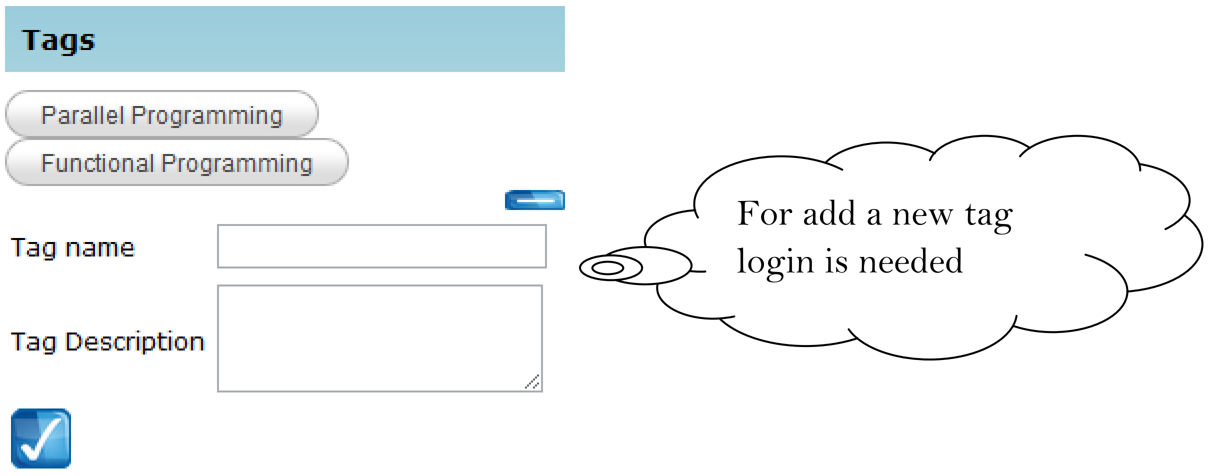
3.4.8 Web 2.0 Features

3.4.8.1 Tag

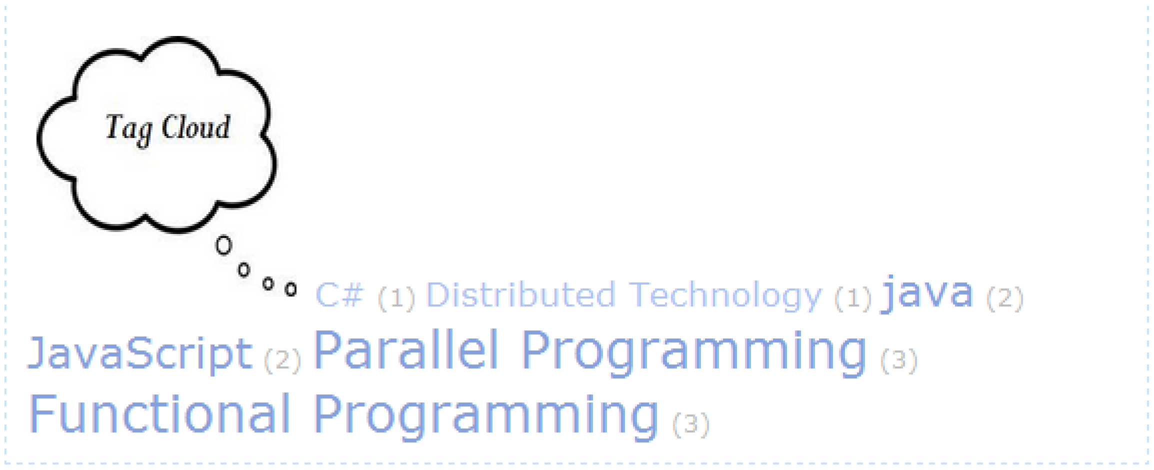
A tag is used to describe a publication in knowledge platform. Tags are a non-hierarchical text data that often are used to help users search for relevant content and usually help to categorization and classification the data content. Front-End make user able to apply a new tag to a publication to allows users to classify their collections of publication in the ways that they find useful.

3.4.8.2 Tag Cloud

Tag cloud is a box containing a list of tags in visual representation. Front-End has used a weighing module to calculate the weight of a particular tag. In weighing module the font size of a tag is determined by its incidence. The frequency of tag is displayed beside the tag.



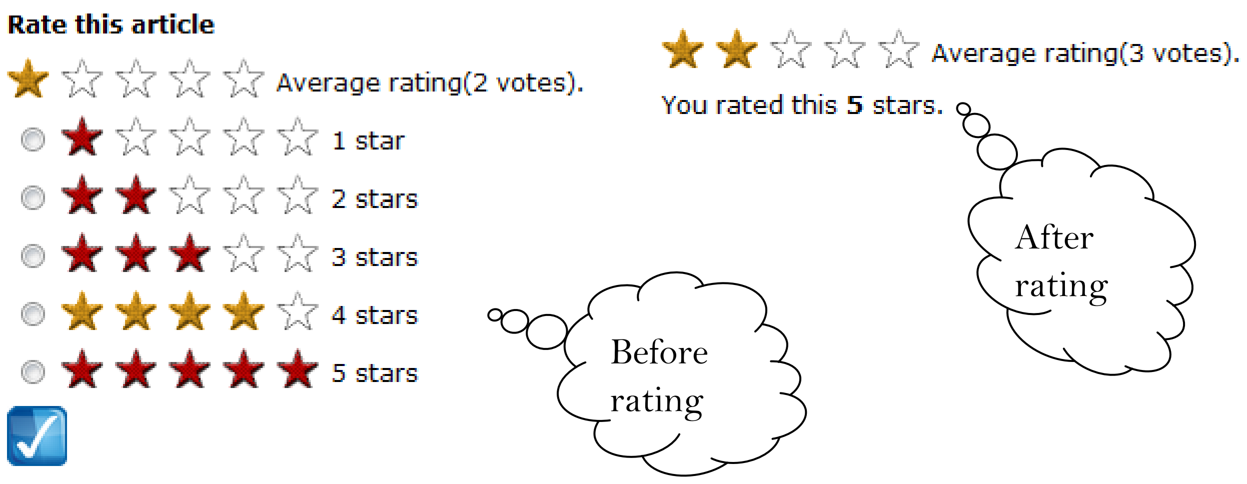
Add new tag in Front-End


Tag Cloud
C# (1) Distributed Technology (1) java (2)
JavaScript (2) Parallel Programming (3)
Functional Programming (3)

Tag cloud in Front-End

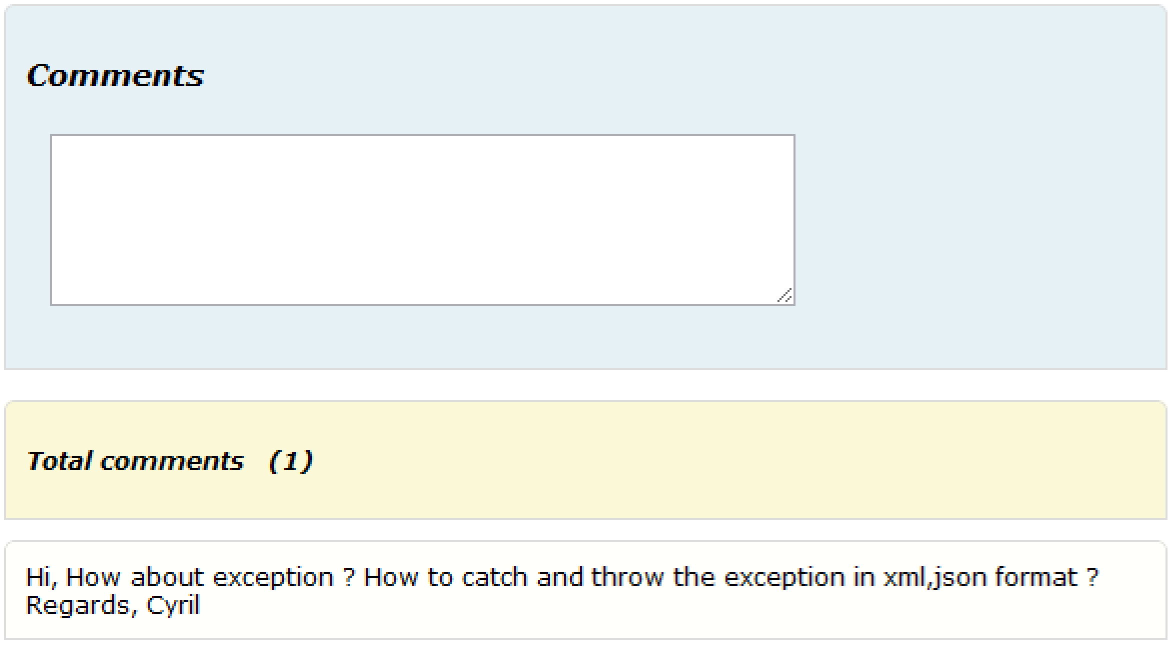
3.4.8.3 User Rating

User rating is a function of Front-End that allows users to rate a specific publication. Front-End's rating systems is “five star” systems. A user rating is used as a means to identify quality of publications based on the views of individual user. Front-End rating is anonymous then ratings can avoid problems such as popularity contests, rating spam, and so on.



3.4.8.4 Comment

Front-End provide for a reader of an article tools to leave comments. For leaving a comment login is needed. Comment makes reader able to share their perspectives, opinions, thoughts and experiences.



Hi, How about exception ? How to catch and throw the exception in xml,json format ?
Regards, Cyril

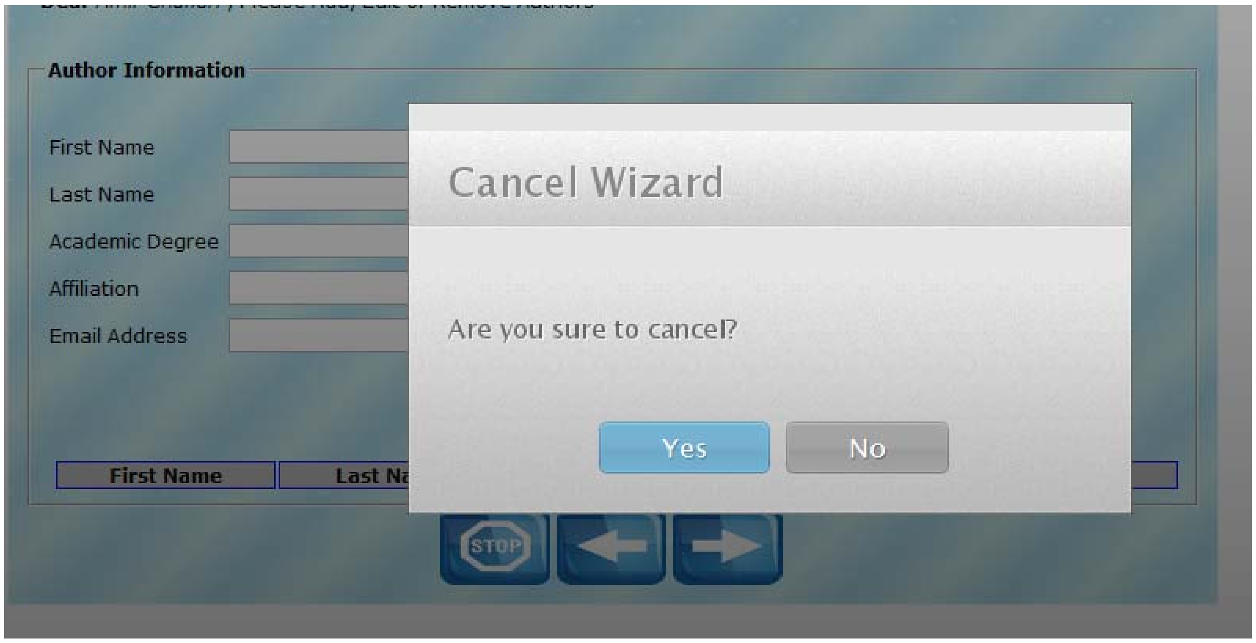
Front-End's commenting facility

3.4.8.5 Desktop Functionality

Providing desktop functionality was one of the basic requirements of Front-End. Front-End have used Ajax and JQuery to address this requirement.

- Confirmation, Modal information message, Tooltip

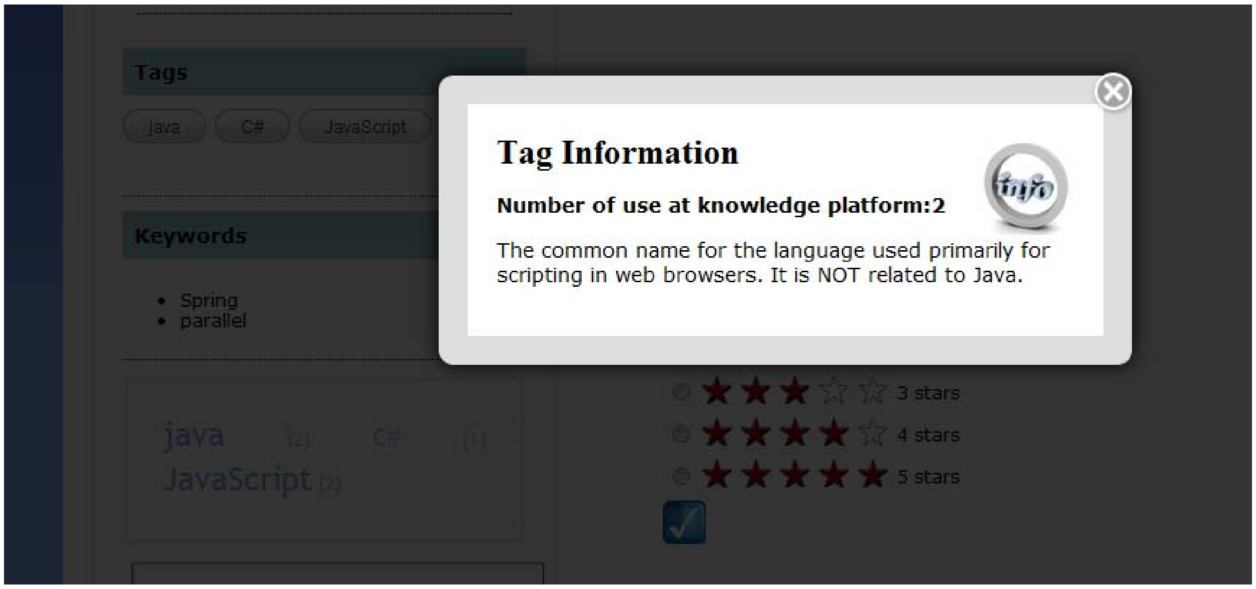
Front-End has applied JQuery to support these features.



Cancel Wizard

Are you sure to cancel?

Using JQuery for modal confirmation

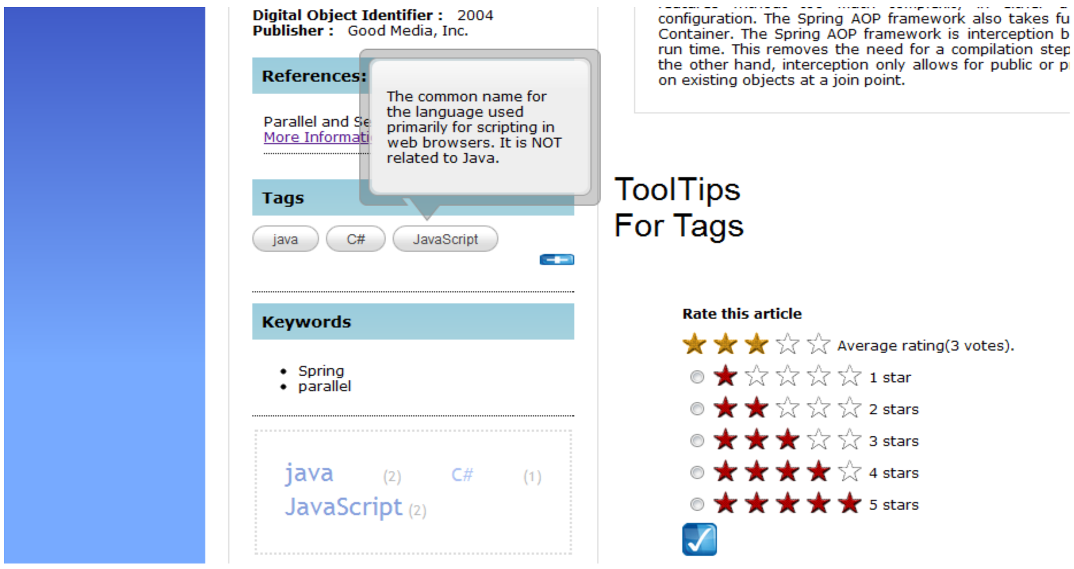


Tag Information

Number of use at knowledge platform:2

The common name for the language used primarily for scripting in web browsers. It is NOT related to Java.

Using JQuery for modal information windows


Digital Object Identifier : 2004
Publisher : Good Media, Inc.

References:

Parallel and Se
More Informati

The common name for the language used primarily for scripting in web browsers. It is NOT related to Java.

Tags

java C# JavaScript

Keywords

- Spring
- parallel

java (2) C# (1) JavaScript (2)

configuration. The Spring AOP framework also takes fu
Container. The Spring AOP framework is interception b
run time. This removes the need for a compilation step
the other hand, interception only allows for public or p
on existing objects at a join point.

ToolTips For Tags

Rate this article

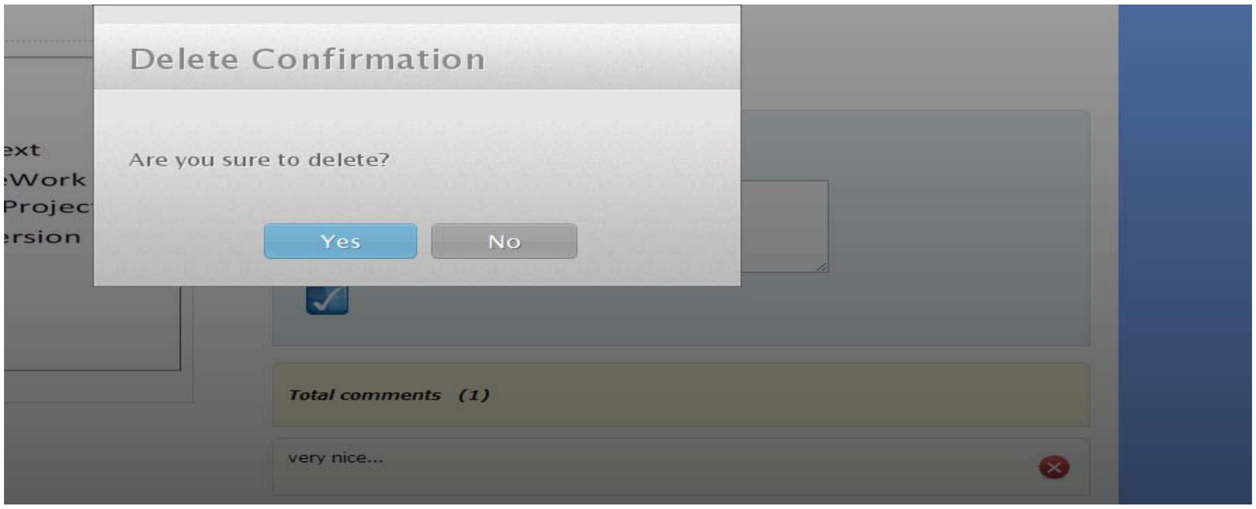
Average rating(3 votes).

1 star
2 stars
3 stars
4 stars
5 stars

Graphical Tooltip by JQuery

- Using Ajax for exchanging data with a server without reloading pages

Front-End has applied Ajax to communicate with server asynchronously without interfering with the displayed page. After receiving data Front-End use JavaScript to modify the Document Object Model (DOM) of the HTML page.


Delete Confirmation

Are you sure to delete?

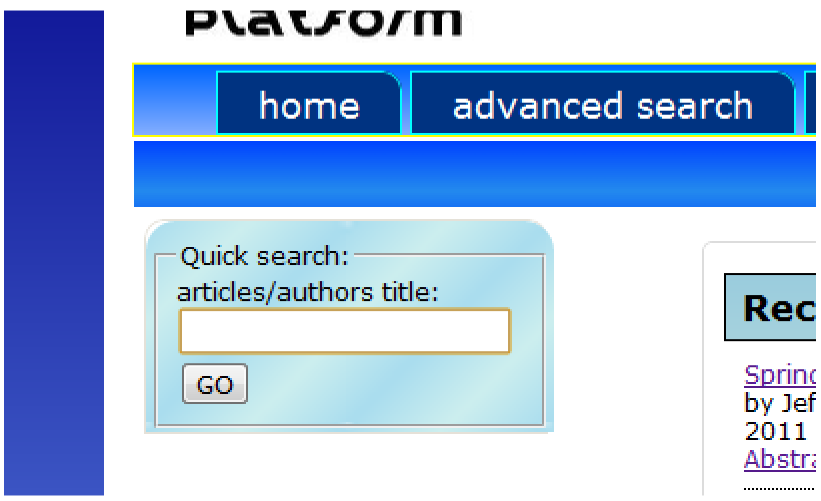
Total comments (1)

very nice...

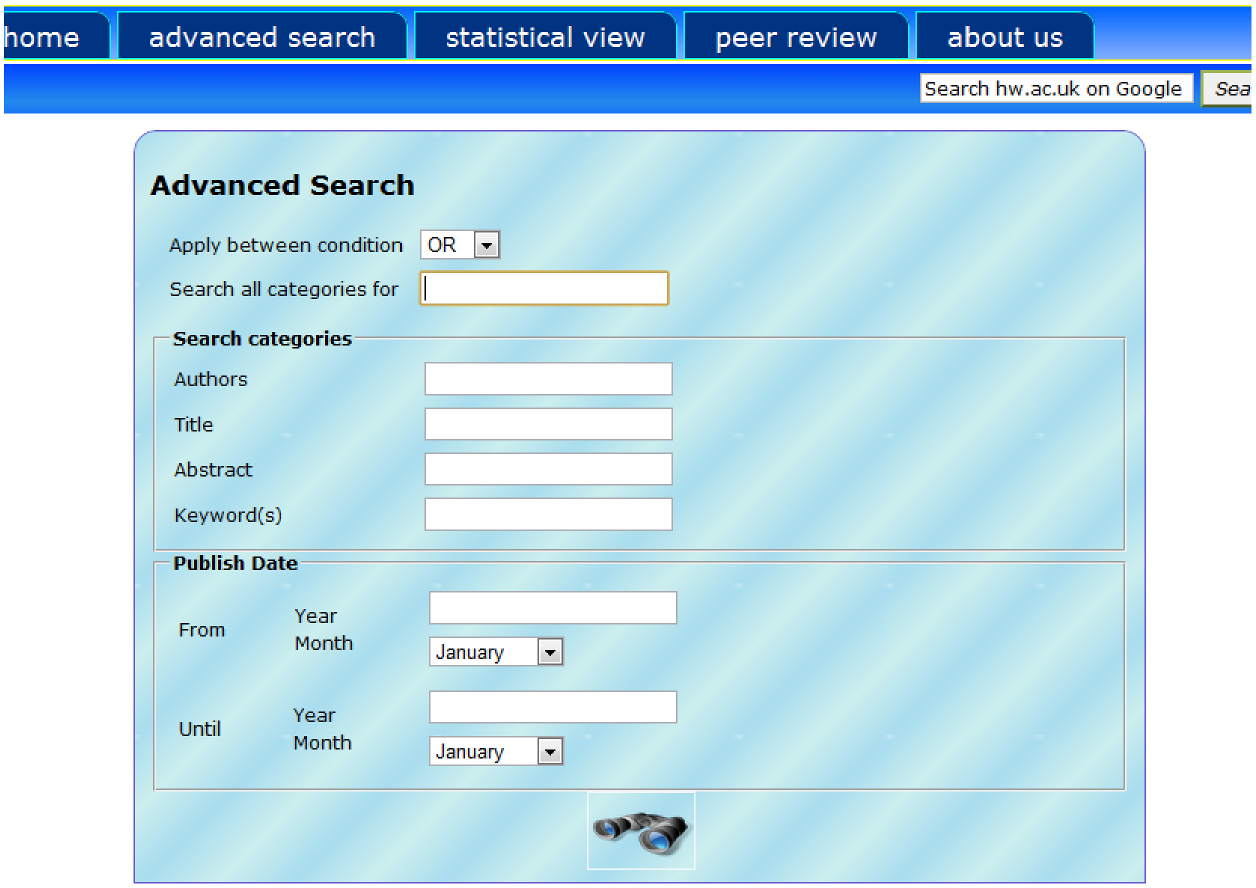
Using Ajax to delete comment

3.4.8.6 Search Publication

One of the most important Front-End's requirements is searching. Searching has a main role in online publication. Front-End provides two different search panels. First one is a quick search that makes users able to search at home page. The quick search panel provides a search on tile of authors and publication.



Since searching publication based on various kinds of data is essential for reader Front-End support an advanced panel to give a more power full option for searching. In advanced search panel user are able to search publication based on author title, publication title, abstract, keywords, and publication's date. There is an option for user to select an operation between search items. Search operation can be AND/OR. By default the operator is OR .



Advanced Search

Apply between condition	OR
Search all categories for	

Search categories

Authors	
Title	
Abstract	
Keyword(s)	

Publish Date

From	Year	
	Month	January
Until	Year	
	Month	January

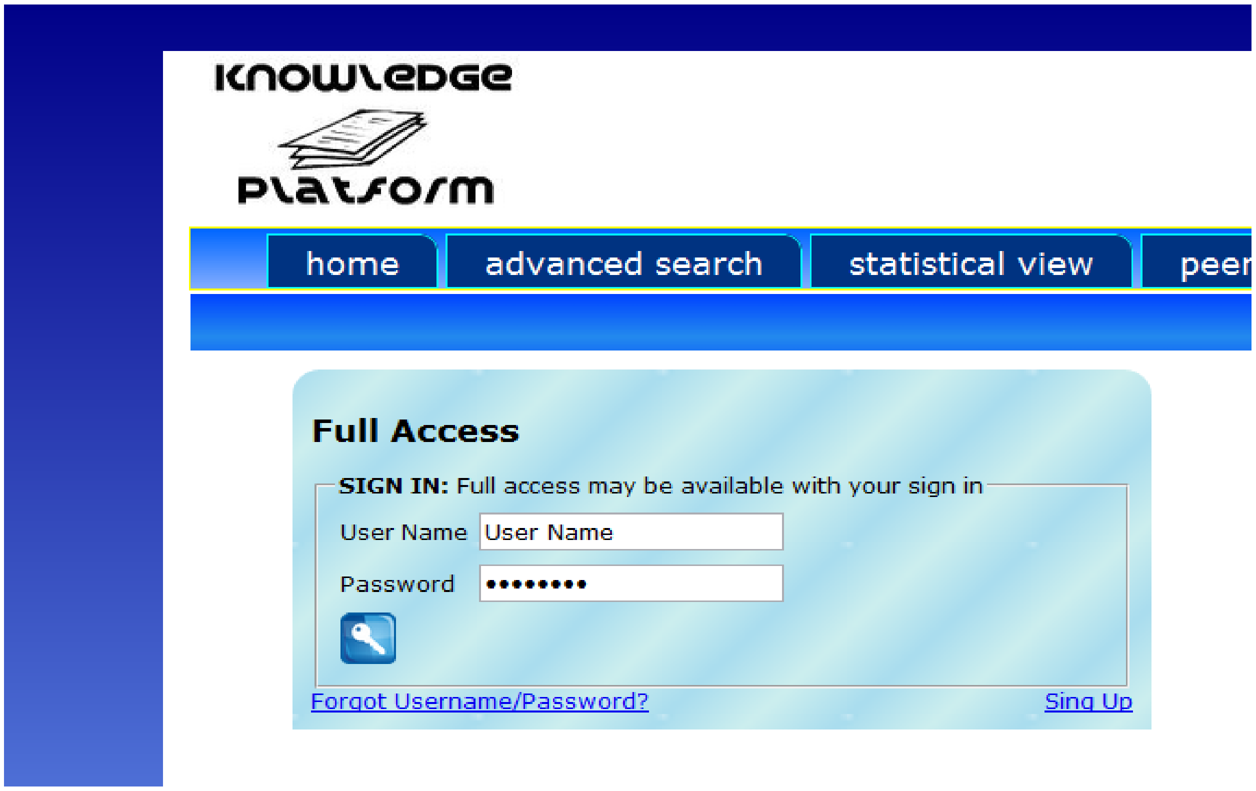
Front-End's advanced search panel

3.4.8.7 Cross Referencing

Front-End uses keywords and tags to implement cross referencing. Front-End's user is able to find all other publications that have specific keyword or tag. Keywords are defined by authors and tags are defined by users. Keywords are more accurate and trustable but tags are also good mechanism for user to customize categorization of information. Core also provides a mechanism to communicate with other digital library such as CiteSeer but due to low-speed transportation of information, Front-End does not integrate it.

3.4.9 User Management

Knowledge Platform supports a role-base user management mechanism. Role-base user management restricts user access to the resources based on their role in the system. In Role base system the permission for certain operation is assign to the roles and users are not assigned permission individually. Role and user definition are done in Core application and Front-End only do login stuff. Detecting user's authorization is done in Core. For accessing to peer review parts and leaving comment, adding new tag login and authentication is necessary.



Full Access

SIGN IN: Full access may be available with your sign in

User Name	User Name
Password	•••••••••

Forgot Username/Password?
Sing Up

Front-End's Login form

When user do the login things on login page and sends user/password to Core for verification a unique session ID is generated by Core. Front-End keeps this unique session ID for each authenticated user. This ID is a user identification that is sent to Core with each request separately.

```
/* Read the session ID from Core and Put it in Http cookie header field for next request */

@Override
public boolean login(Login login, HttpSession session) {
    if(!Util.isLocalData(session)){
        HttpEntity<Login> entity  = new HttpEntity<Login>(login);
        ResponseEntity<Login> response=null;
        try{
            response = restTemplate.postForEntity(Login.URL, entity, Login.class);
            Login result = response.getBody();
            HttpHeaders headers = response.getHeaders();
            String sessionId=Util.getHeaderValue("set-cookie",headers);
            if(sessionId==null||sessionId.trim().isEmpty()) {
                session.setAttribute(Util.LAST_LOGIN_STATUS, Util.LOGIN_STATUS_FAILD);
                return false;
            }
            sessionId=sessionId.substring(0,sessionId.indexOf(";"));
            session.setAttribute(Util.USER_SESSION_ID, sessionId);
            session.setAttribute(Util.LAST_LOGIN_STATUS, Util.LOGIN_STATUS_SUCCESSFUL);
            login.setUserAtom(result.getUserAtom());
            User user = userWebService.get(User.class, login.getUserAtom().getHref(), session, null);
            login.setUser(user);
            session.setAttribute(Util.USER_SESSION,user);
            return true;
        }
        catch(RestClientException ex){
            session.setAttribute(Util.LAST_LOGIN_STATUS, Util.LOGIN_STATUS_FAILD);
            System.out.println("RestClientException in Login:"+ex.getMessage());
            if(ex instanceof HttpServerErrorException){
                HttpServerErrorException httpServerErrorException=(HttpServerErrorException)ex;
                throw new IllegalStateException(httpServerErrorException.getResponseBodyAsString());
            }
            return false;
        }
        catch(Exception ex){
            session.setAttribute(Util.LAST_LOGIN_STATUS, Util.LOGIN_STATUS_FAILD);
            System.out.println("Rest Call Exception in Login:"+ex.getMessage());
            return false;
        }
    }
    else{
        return SampleData.getLocalData(session).login(login, session);
    }
}
```

A sample code of Front-End that get session ID from Core

Chapter 4

Evaluation

In this chapter the Front-End's evaluation process is described. The evaluation's purpose and the methods used to collect and analyse data are reviewed. In order to evaluate the Front-End an online survey was conducted. Evaluation questions enable us to better assess the usability of the Front-End and the results are used to improve the Front-End functionality. Moreover, some parts of evaluation's results specify future research. The questionnaire was used can be found in the appendix B.

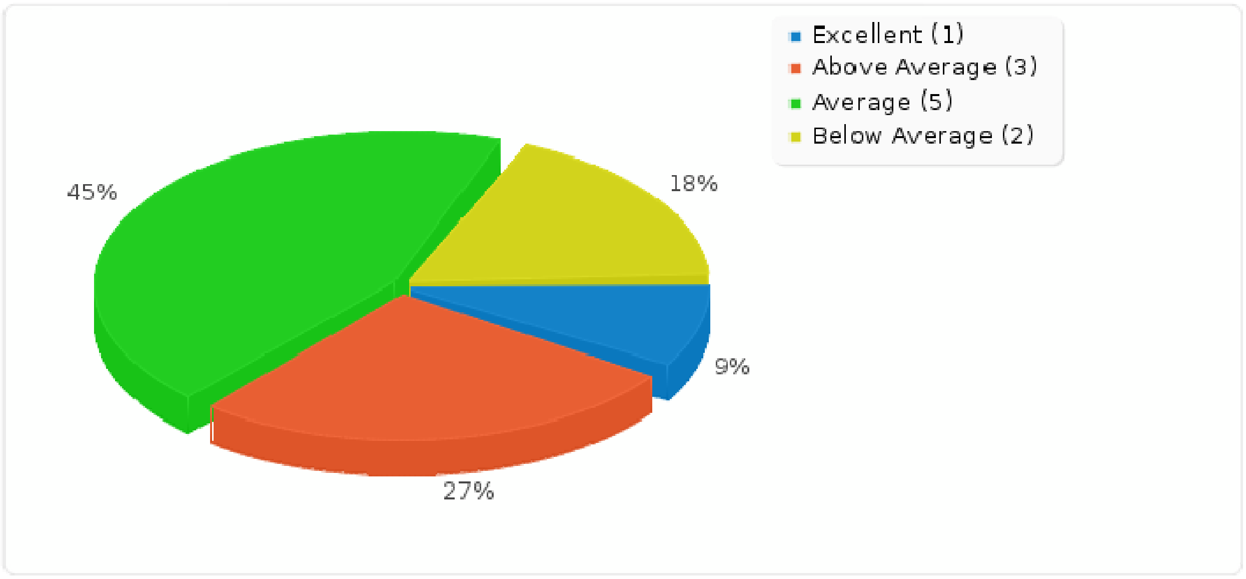
4.1 Evaluation Plan

The Front-End has been evaluated with regard to two aspects. First, web 2.0 technologies were evaluated in Front-End. Since one of the most important requirements of this project was using the web 2.0 technologies to bring new possibilities to interaction and provide the best possible user experience. Secondly, the peer review process which is another project requirement was evaluated. The questionnaire was designed to meet the above objectives.

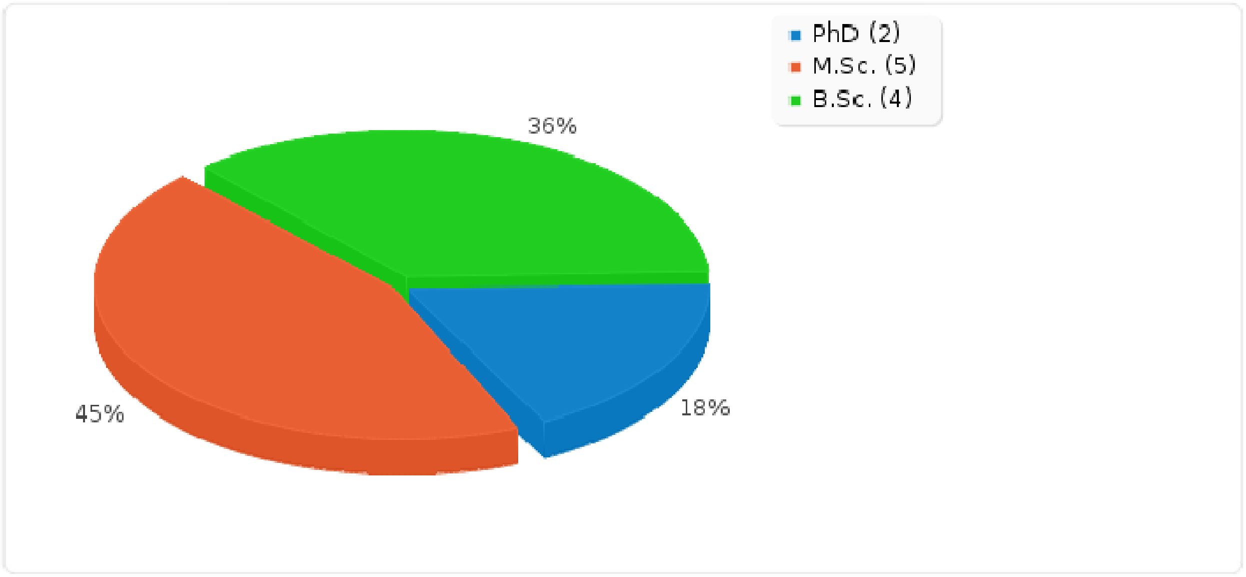
4.2 Description of Evaluation Method

During the evaluation phase eleven people filled out the online survey. Evaluation was done online. Participants in evaluation have at least Bachelor degree because an academic study is necessary to have a minimum knowledge about scientific publication but most of participant have a master's degree or above. In response to question "How much are you familiar with peer review process?" the result shows that only 18 percent of participants selected "below average" option. The result shows that we can rely on the participant's feedback.

Also about 63 percent of participants have a master's degree or above that shows they have acceptable knowledge about scientific publication.



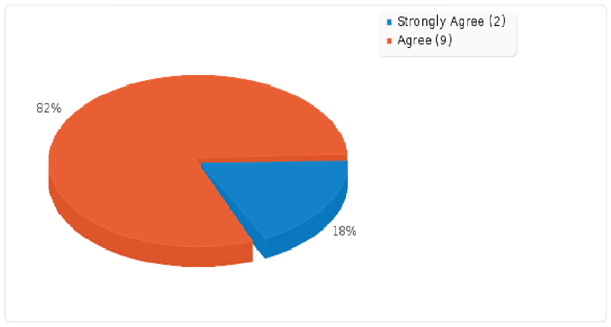
How much are you familiar with peer review process?



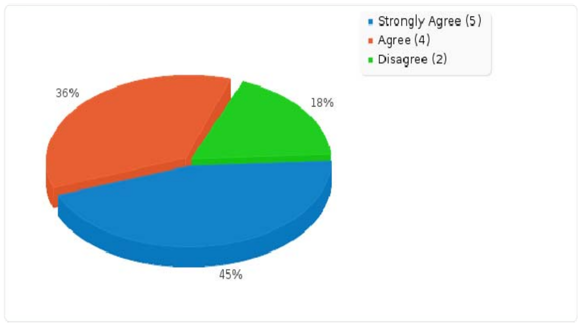
What is your last qualification?

4.3 Evaluation Discussion

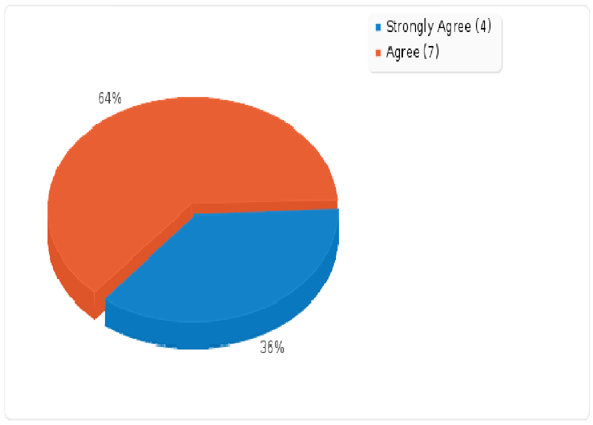
Firstly we discuss the result of evaluation of Web 2.0 technologies (such as tagging, linking, search mechanism, cross referencing) in Front-End. These aspect results are listed as follows:



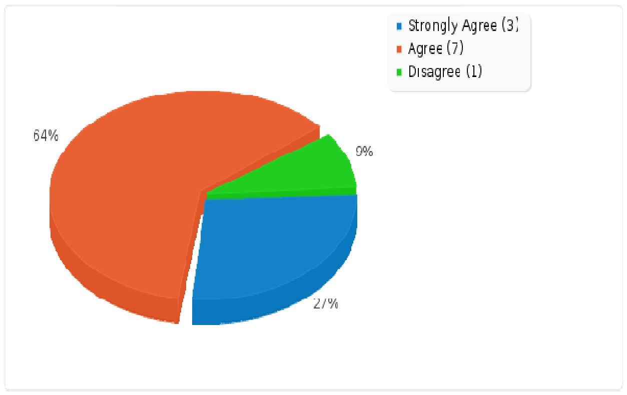
(1)



(2)



(3)



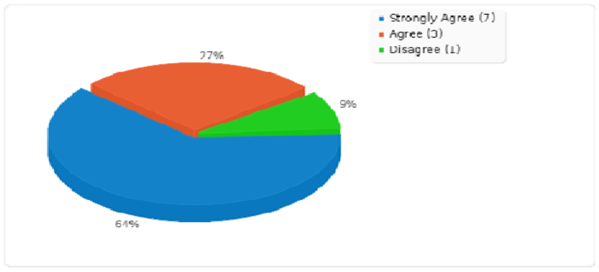
(4)

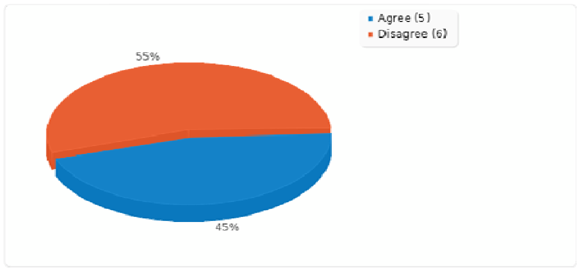
1) Front-End provides a good mechanism for categorization of content by users (Such as tagging and tag cloud features)
2) Front-End provides a user friendly environment for exchanging views and collaborating. (Comment, rating, vote)
3) Front-End provides a good mechanism for finding information through keyword and tag searching.
4) Front-End support easy-to-use features for linking and referencing of information. (Such as using hyperlink for finding more info about authors, references, keywords, citation)

As it is clear from the result, search facilities have the best satisfaction percentage. Second satisfaction level is for Content Categorization (such as tagging, and tag cloud). The third one is linking and referencing of information. The last one is exchanging views and collaborating such as rating, commenting and voting.

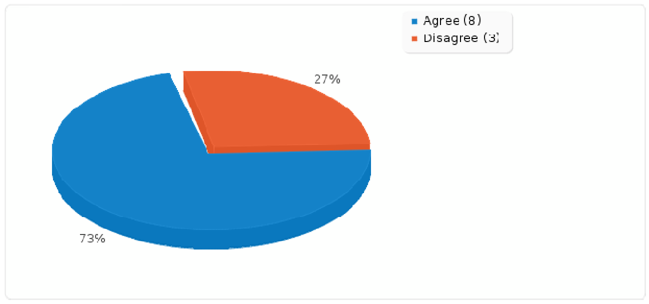
But in an overall view most of the users are satisfied with Web 2.0 technologies. Search facilities such as advanced search panel, tagging and keyword bring user satisfaction.

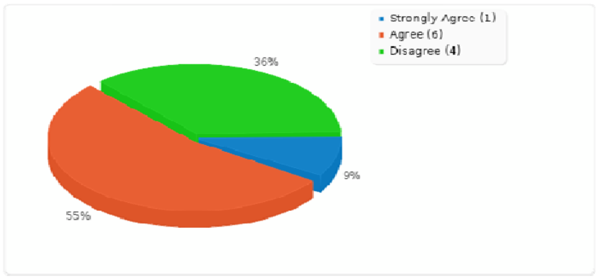
However, evaluation result for peer review is different than web 2.0 result.





(1) (2)





(3) (4)

1) Front-End provides a good mechanism for authors to submit manuscripts online. (Upload manuscripts/supplementary files, Meta data such as title, abstract, authors)
2) Front-End supports a good tracking system for authors during publishing process.
3) Front-End provides a good Editorial board mechanism. (Assigning an article to reviewer, monitor manuscript status)
4) Front-End supports a good Reviewer tracking system.(Such as notification, response template)

From the result, can be concluded that Editorial board mechanism have the best satisfaction percentage. Second satisfaction level is for authors tracking. The third one is author's submission manuscripts. The last one is Reviewer tracking system.

The low satisfaction in peer review can have some reasons. First of all, peer review procedure is complicated and there are many business logics that need more time to test and improvement. Moreover, Peer review does not have a determined procedure and there are many different definition of peer review process. Finding a general and common solution for peer review process would bring user satisfaction.

Chapter 5

Conclusion

This chapter presents the achievements and limitations for this project. In Section (5.1) the limitations of the project is discussed. Section (5.2) outlines the further research and work. Section (5.3) presents the conclusions and achievements.

5.1 Summary

Is knowledge platform an invention?

This is an important question that each research should be able to response it. In this part, we are going to answer this question and also find appropriate name for our invention.

The main characteristic of knowledge platform that makes it different from every other platform is use of web-based technologies to turn communication into an interactive dialogue. Knowledge platform tries to mix existing phenomenon such as Searching, linking, Authoring, Tagging, Blogging into a scientific platform to provide a secure and user-friendly environment for scholarly community.

This project's aims was to show how we can mix a social networking's features such as communicate with text-based comments, good search functionality, free membership, navigation links, rating and tagging with current scholarly community/publication such as digital library(CiteSeer) or peer review journals (ACM, IEEE).

Mixing these two features makes Knowledge platform a different from every other platform. This project brings a new concept for scientific publication and scholarly community that we can name it "*a social network for scholarly community*".

5.1.1 The Knowledge Platform 2.0 Effort

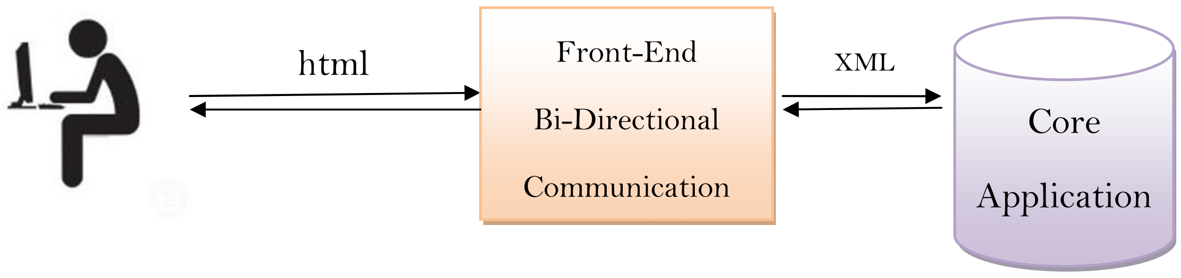
Knowledge platform consist of two parts. First part is Core and the second one is Front-End. Core plays role in providing required data for Front-End. Front-End communicate with Core via REST web service. Front-End is more involved in providing a user friendly interface for users. Separating the duties of publishing data bring some benefits to final solution. By separating duties each part has more opportunities to concentrate more on finding and using the last technologies in its domain. During duties separation, Front-End was responsible to find and use latest features of web 2.0 technologies that can improve user's contribution during the knowledge dissemination process. In next section we will discuss how Front-End applied current features in disseminating and publishing scholarly knowledge.

5.1.2 Front End

Front-End have tried to discover the latest web 2.0 technique in other domains such as social networks and blogs to find the latest technologies that are applied in that domains. After discovering these features like tagging, rating, commenting Front-End try also to apply the last technologies in this field. One of the most powerful, understandable, and universal concept for improving role of these features is rich internet application (RIA) concept. Front-End use that aspect of RIA that does not need to install a plug-in on client side. Concepts like Ajax, JQuery, JavaScript, DOM are standards that are supported in all platforms and browsers. Front-End use these aspects of RIA because these technologies are not dependent on specific plug-in or platform. Dependency to a specific platform is a weak point for a wide access application like Knowledge Platform. Using other kinds of RIA such as Adobe Flash, JavaFX, and Microsoft Silverlight makes Front-End dependant to specific plug-in.

5.1.3 Bi-Directional Communication

One of a good experience that was gained during the development process of Front-End is bi-directional communication. Front-End plays two roles simultaneously. As a user's view, it is a server application that provides desktop functionality for Knowledge platform's users. In this view, Front-End is an enterprise Java web application that provides dynamic pages for users. On the other hand, Front-End is a client application for Core. For playing this role Front-End use Spring client side Restful library that provides good facilities for a client application. During this process Front-End's user can not recognize that Front-End is a client application.



5.1.4 Project Evaluation

As it was discussed before in evaluation chapter Front-End's web 2.0 feature was evaluated by participants. In an overall view most of the users were satisfied with Web 2.0 technologies especially search facilities such as advanced search panel, tagging and keyword. We can conclude from the result of evaluation that web 2.0 techniques bring users satisfactions especially tag and tag cloud that makes user categorization easier for uses and it also provides cross referencing.

5.2 Limitation

Knowledge Platform is an enterprise project and Front-End also is enterprise project. For connectivity between Core and Front-End REST the web service that has been used is not a good choice. SOAP includes a series of service characteristics like priority, expiration, security credentials, routing information and transactional behaviors that make it suitable for enterprise project. Moreover it is a clear rule that we should try **to reduce the communication cost** on the distributed system but using atom link by Core increase incredibly communication time. For Example to do a simple fetch (fetching a publication's info) we need to send lots of separated requests for only one publication and all its dependency (dependency such as Tags, Comments, Keywords, Authors, etc). In this method if a publication has 5 tags, 10 keywords, 3 authors and 2 comments Front-End should send 20 separated request to fetch only one publication's information. Using this method between two systems causes a bad affect on system performance.

Moreover, Peer review procedure has many complicated rules and business logics. However, due to the time limitations for this project (about 3 months) these logics and rules are not completed and during evaluation there are many suggestions that can improve peer review procedure. During Front-End's development process I have used the best available technologies such as Spring framework, Hibernate validation, Apache upload file. I have spent too much time to learn these technologies. Moreover, consistency with Core project takes many time then we could not spend enough time for finding a general solution for peer review procedure. Therefore, in evaluation, peer review has not got a high satisfaction.

5.3 Future Work

During the evaluation a number of aspects were found that need to be improved. Some of these aspects are related to peer review process. Other aspects are related to performance. A good tracking system for submitted manuscript is one of the most important parts of peer review. After submitting a manuscript there are many mechanisms that can improve tracking system such as automatic email, an alarm system for referee's deadline date. Developing a monitoring system for editorial board to trace the manuscript during peer review procedure can be a good idea for future extension.

By using SOAP web service instead of REST web service and designing a lower cost method for exchanging information(we have use atom link that bring a high communication cost) we can improve the system's functionality and consequently it will bring a higher user's satisfaction.

Professional, Ethical and legal Issues

- Professional issues
  During survey the current system fairness is essential. Results of survey may not be a true and fair conclusion of current knowledge platform if there will be personal biases or discrimination against or in favor of a system.
  Also users provide through registration some personal information for us. The information is not used for any other purpose. We keep secure what information user enter in our web application by a one-way encryption method.
  Knowledge platform use a role-base mechanism for accessing to the submitted information. Access to the publications that are submitted for peer review process is controlled by user's role. In each stage based on peer review procedure only permitted user such as editor, reviewer, author have access to the publication and author's information.
- Legal issues
  There is not legal issue to be considered in this project. All sources that are discussed in project are open source. We try to find positive points of different system to put the best method in this project. All considered systems and technique are open source and free to change and distribution. Developing tools in the project such as database engine (MySql) and programming IDE are also open source and free to download and use.
- Ethical issues
  There is not any data or materials that need permission in this research. Also the research does not involve human subjects then evaluation form is anonymous. Then all research methods and information are open to public scrutiny and can be seen to be subject to the highest ethical standards.
- Front-End is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

References

1. *Web and Electronic Publishing Trends.* **Mohan, H.R.** University of Mysore : Joint Workshop on Digital Libraries, 2001.

2. **Gottfried Vossen, Stephan Hagemann.** *Unleashing Web 2.0.* s.l. : Denise E. M. Penrose, 2007. 978-0-12-374034-2.

3. *Citation-based journalranks:The use of fuzzy measures.* **SimonJames, Gleb Beliakov-.** Burwood : Elsevier North-Holland, Inc, 2010. 0165-0114.

4. *Electronic Publishing.* **Peter B. Boyce, Evan Owens, and Chris Biemesderfer.** s.l. : American Astronomical Society (AAS), 1997.

5. *Digital Libraries and Autonomous Citation Indexing.* **Steve Lawrence, C. Lee Giles, Kurt Bollacker.** s.l. : IEEE, June 1999. 0018-9162.

6. *HyperJournal, PHP scripting and Semantic Web technologies for the Open Access.* **Michele Barbera, Francesca Di Donato, Giovanni Tummarello, Christian Morbidoni.** Pisa, Italy : Università Politecnica delle Marche.

7. *DiVA.* **Centre, Electronic Publishing.** s.l. : Uppsala University Library, 2006.

8. *Basics of Research Paper Writing and Publishin.* **Derntl, Michael.** s.l. : University of Vienna-Faculty of Computer Science, May 27, 2003.

9. *AJAX Introduction.* **Schools, W3.** s.l. : W3Schools.

10. *Designing Rich Internet Applications.* **MSDN.** s.l. : microsoft.

11. *HTML DOM Introduction.* **W3C.** s.l. : http://www.w3schools.com/htmldom/dom_intro.asp.

12. *Resource Description Framework.* **Group, RDF Working.** s.l. : http://www.w3.org/RDF/.

13. *The Semantic Web: The Roles of XML and RDF.* **STEFAN DECKER, SERGEY MELNIK, FRANK VAN HARMELEN, DIETER FENSEL, D**

MICHEL KLEIN, JEEN BROEKSTRA, MICHAEL ERDMANN, IAN HORROCKS. s.l. : IEEE Internet Computing, 2000. 1089-7801.

14. *Interoperability for Digital Objects and Repositories.* **Sandra Payette, Christophe Blanchi, Carl Lagoze, Edward A. Overly.** s.l. : D-Lib Magazine, May 1999. 1082-9873.

15. **Inderjeet Singh, Beth Stearns,Mark Johnson, and the Enterprise Team.** *Designing Enterprise Applications with the J2EE Platform, Second Edition.* 2002. 0-201-78790-3.

16. **Bryan Basham, Kathy Sierra, and Bert Bates.** *Head First Servlets and JSP.* s.l. : O'Reilly Media, Inc., 2008. 978-0-596-51668-0.

17. **Rod Johnson, Juergen Hoeller, Keith Donald, Colin Sampaleanu, Rob Harrop, Alef Arendsen, Thomas Risberg, Darren Davison, Dmitriy Kopylenko, Mark Pollack, Thierry Templier, Erwin Vervaet, Portia Tung, Ben Hale, Adrian Colyer, John Lewis, Costin Leau, Mark.** Reference Documentation. *Spring Framework.* [Online] 2004-2010 . http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/index.html.

18. *objectsource.* [Online] http://www.objectsource.com/j2eechapters/Ch19-I18N_and_L10N.htm.

19. *online tutorial-Implementing OAI-PMH.* **Forum, Open Archives.** s.l. : http://www.oaforum.org/tutorial/english/page4.htm.

20. *Model-View-Controller Pattern.* **enode.** s.l. : http://www.enode.com/x/markup/tutorial/mvc.html.

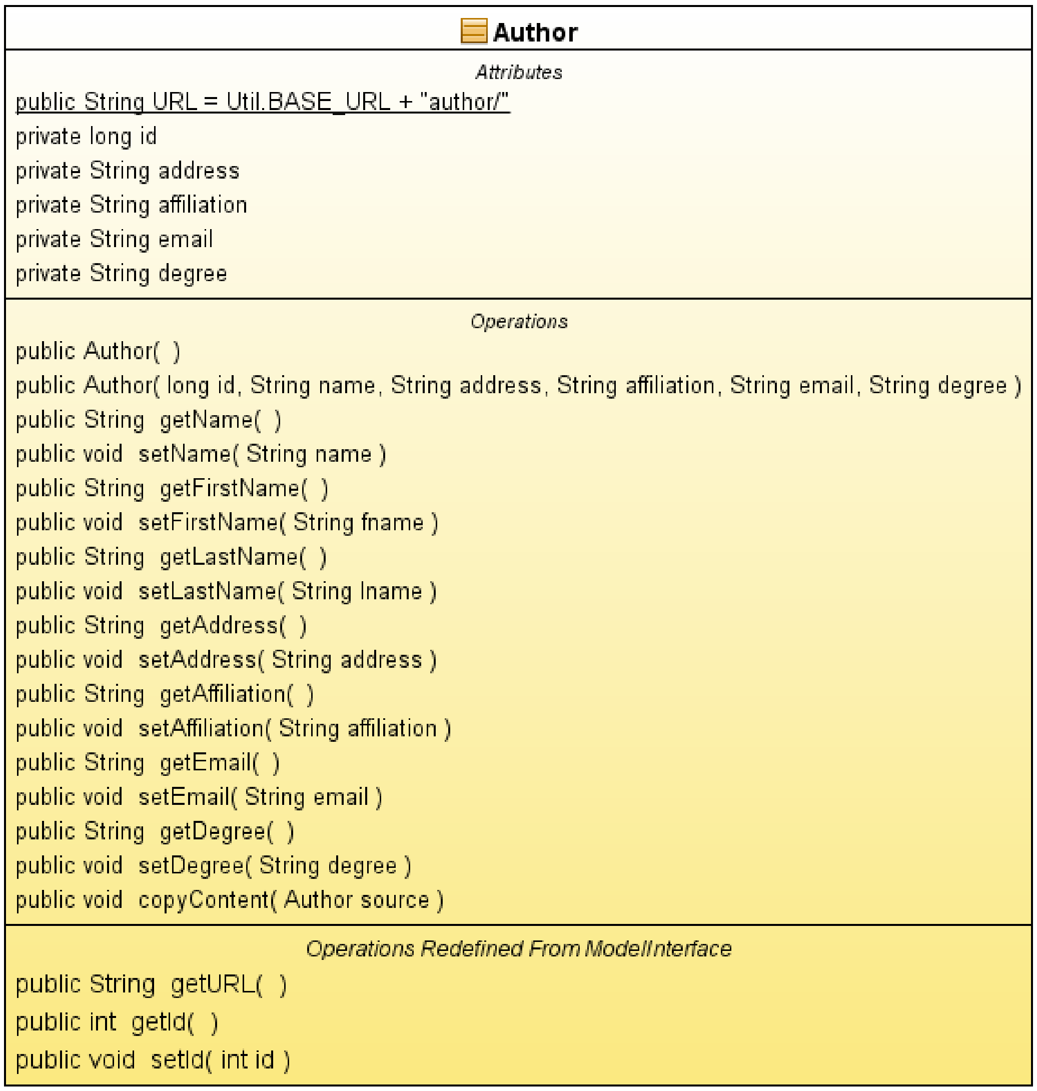
21. *Interoperability and Open Archives Initiative Protocol for Metadata* . **Martha Latika Alexander, J N Gautam.** New Delhi : INFLIBNET Centre, 2004.

22. **Hackett, Scott.** plug-in-extensibility-through-reflection-in-net-and-java/. *http://blog.slickedit.com.* [Online] Feb 2008.

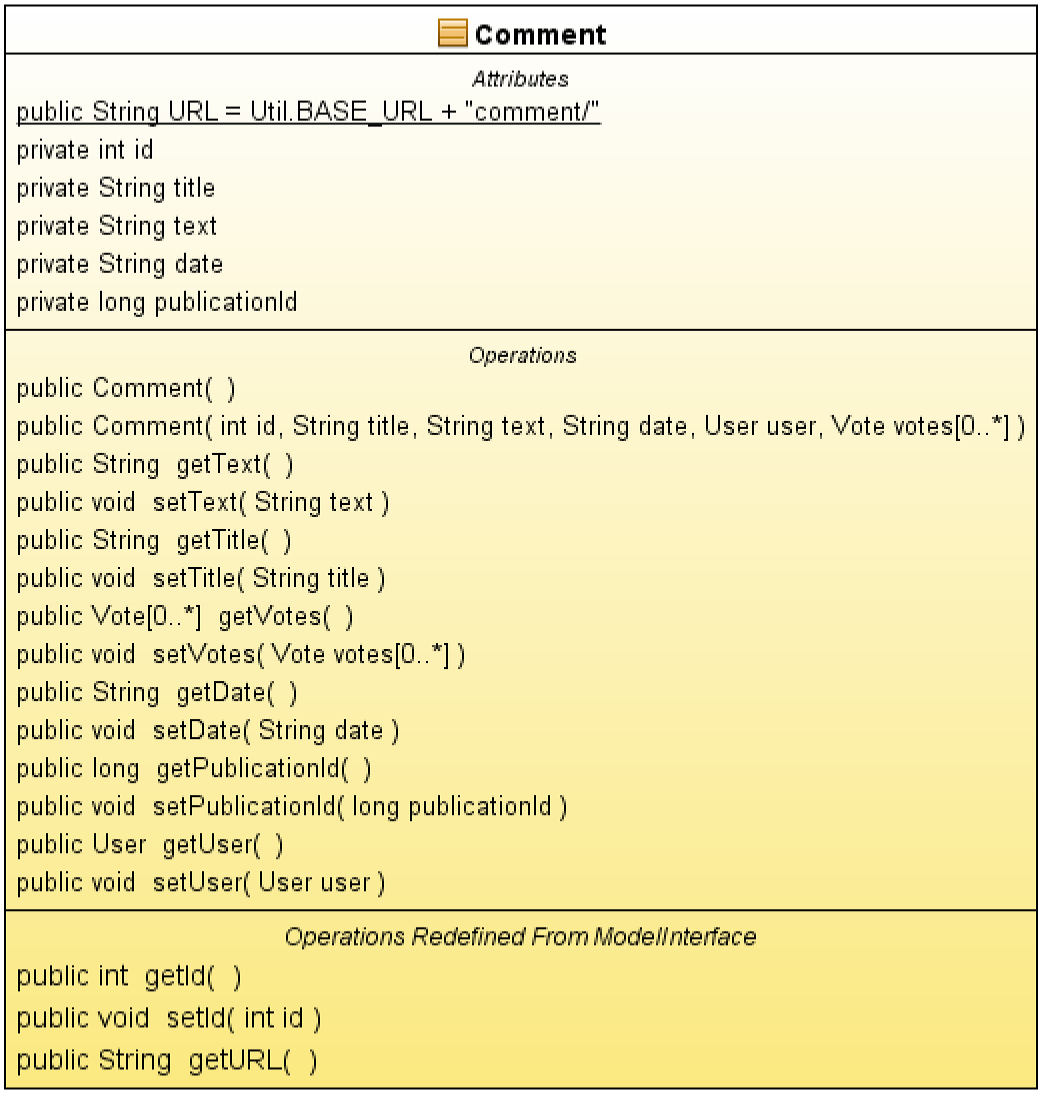
Appendices:

Appendix A

Data Model Classes



```
Author
Attributes
public String URL = Util.BASE_URL + "author/"
private long id
private String address
private String affiliation
private String email
private String degree
Operations
public Author( )
public Author( long id, String name, String address, String affiliation, String email, String degree )
public String  getName( )
public void  setName( String name )
public String  getFirstName( )
public void  setFirstName( String fname )
public String  getLastName( )
public void  setLastName( String lname )
public String  getAddress( )
public void  setAddress( String address )
public String  getAffiliation( )
public void  setAffiliation( String affiliation )
public String  getEmail( )
public void  setEmail( String email )
public String  getDegree( )
public void  setDegree( String degree )
public void  copyContent( Author source )
Operations Redefined From ModelInterface
public String  getURL( )
public int  getId( )
public void  setId( int id )
```



Comment

Attributes

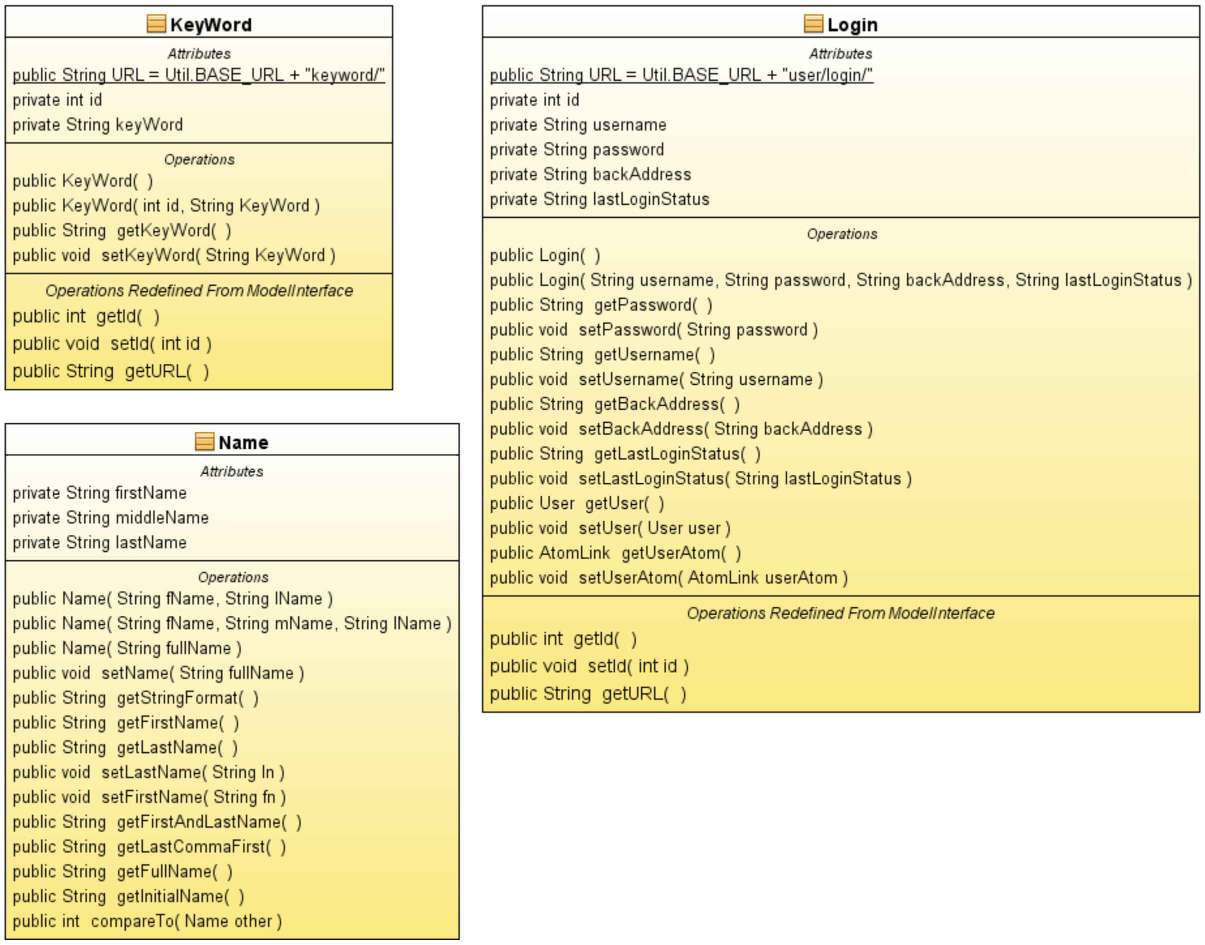
public String URL = Util.BASE_URL + "comment/"
private int id
private String title
private String text
private String date
private long publicationId

Operations

public Comment( )
public Comment(int id, String title, String text, String date, User user, Vote votes[0..*])
public String getText( )
public void setText(String text)
public String getTitle( )
public void setTitle(String title)
public Vote[0..*] getVotes( )
public void setVotes(Vote votes[0..*])
public String getDate( )
public void setDate(String date)
public long getPublicationId( )
public void setPublicationId(long publicationId)
public User getUser( )
public void setUser(User user)

Operations Redefined From ModelInterface

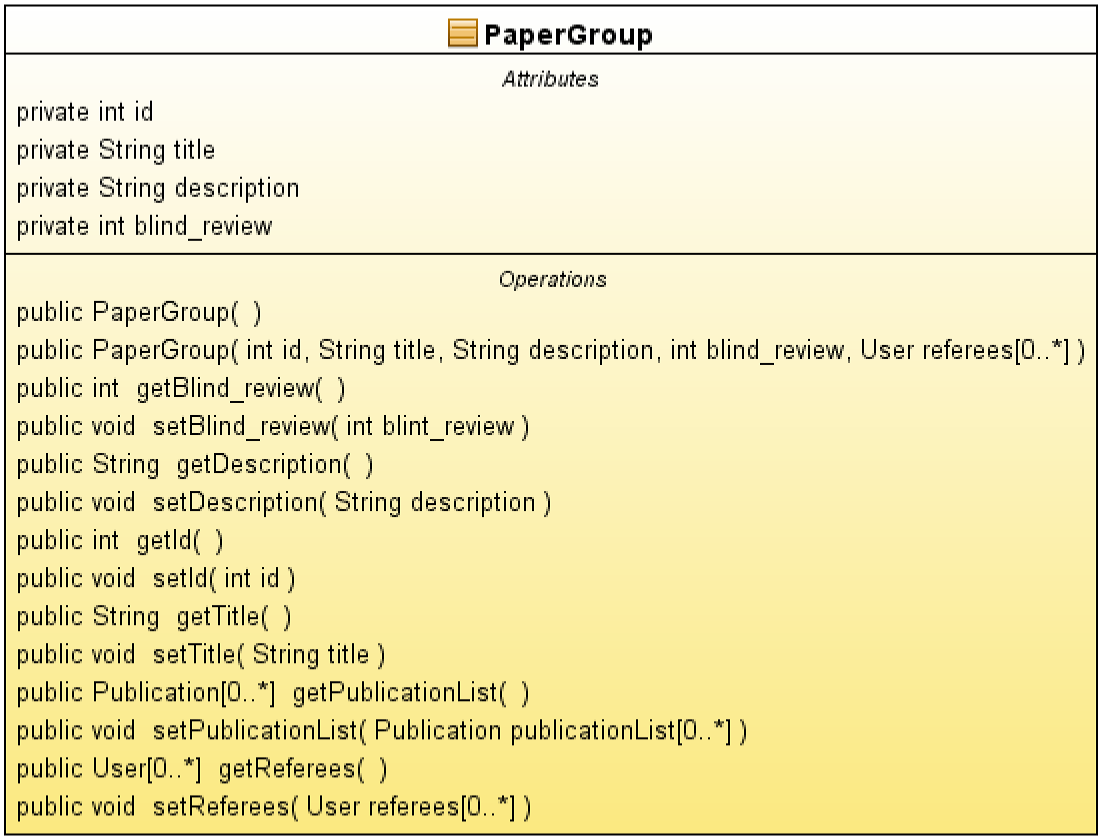
public int getId( )
public void setId(int id)
public String getURL( )



```
KeyWord
Attributes
public String URL = Util.BASE_URL + "keyword/"
private int id
private String keyWord
Operations
public KeyWord( )
public KeyWord( int id, String KeyWord )
public String  getKeyWord( )
public void  setKeyWord( String KeyWord )
Operations Redefined From ModelInterface
public int  getId( )
public void  setId( int id )
public String  getURL( )
```

```
Name
Attributes
private String firstName
private String middleName
private String lastName
Operations
public Name( String fName, String lName )
public Name( String fName, String mName, String lName )
public Name( String fullName )
public void  setName( String fullName )
public String  getStringFormat( )
public String  getFirstName( )
public String  getLastName( )
public void  setLastName( String ln )
public void  setFirstName( String fn )
public String  getFirstAndLastName( )
public String  getLastCommaFirst( )
public String  getFullName( )
public String  getInitialName( )
public int  compareTo( Name other )
```

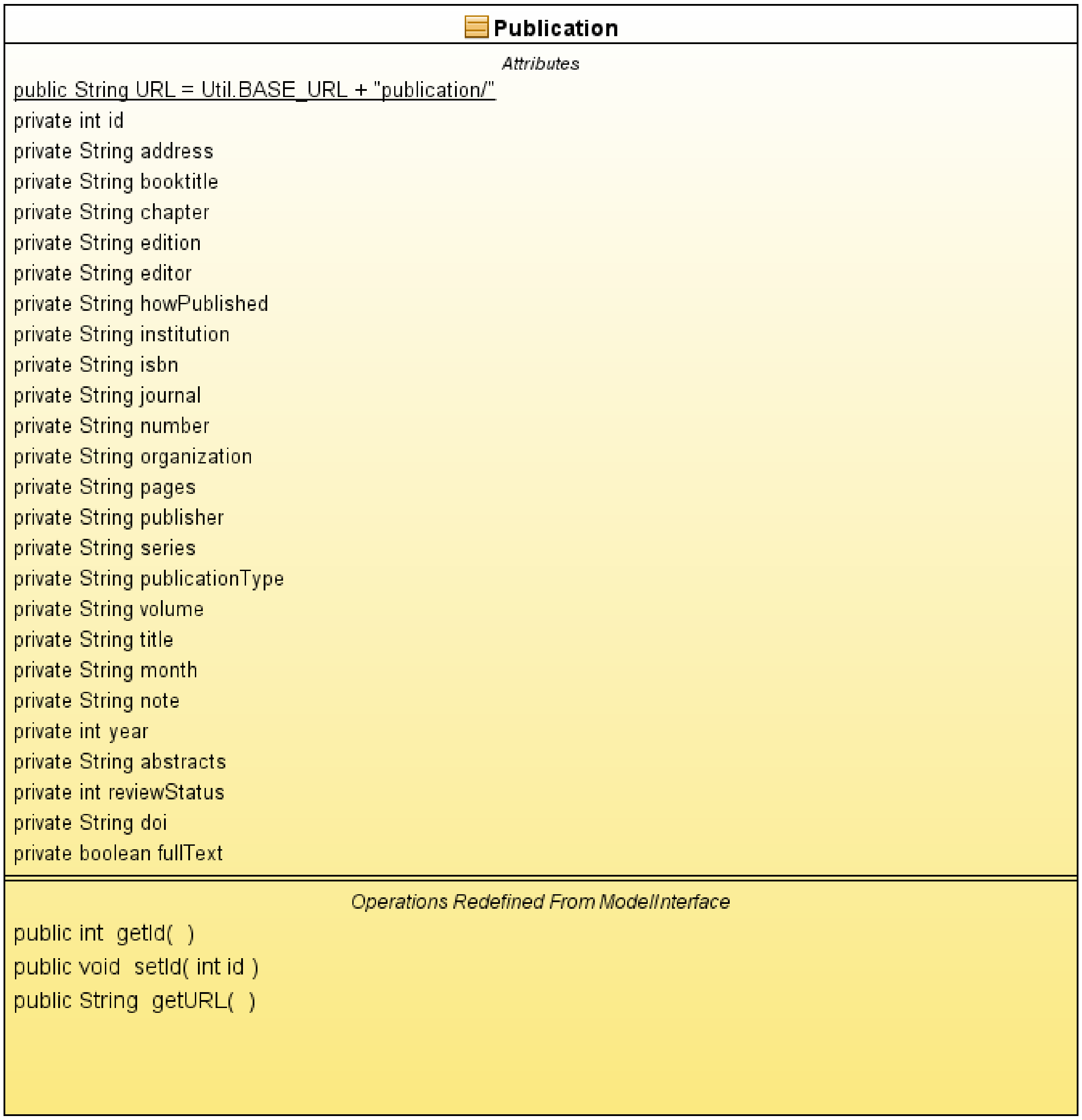
```
Login
Attributes
public String URL = Util.BASE_URL + "user/login/"
private int id
private String username
private String password
private String backAddress
private String lastLoginStatus
Operations
public Login( )
public Login( String username, String password, String backAddress, String lastLoginStatus )
public String  getPassword( )
public void  setPassword( String password )
public String  getUsername( )
public void  setUsername( String username )
public String  getBackAddress( )
public void  setBackAddress( String backAddress )
public String  getLastLoginStatus( )
public void  setLastLoginStatus( String lastLoginStatus )
public User  getUser( )
public void  setUser( User user )
public AtomLink  getUserAtom( )
public void  setUserAtom( AtomLink userAtom )
Operations Redefined From ModelInterface
public int  getId( )
public void  setId( int id )
public String  getURL( )
```



```
PaperGroup

Attributes
private int id
private String title
private String description
private int blind_review

Operations
public PaperGroup(  )
public PaperGroup( int id, String title, String description, int blind_review, User referees[0..*] )
public int  getBlind_review(  )
public void  setBlind_review( int blint_review )
public String  getDescription(  )
public void  setDescription( String description )
public int  getId(  )
public void  setId( int id )
public String  getTitle(  )
public void  setTitle( String title )
public Publication[0..*]  getPublicationList(  )
public void  setPublicationList( Publication publicationList[0..*] )
public User[0..*]  getReferees(  )
public void  setReferees( User referees[0..*] )
```



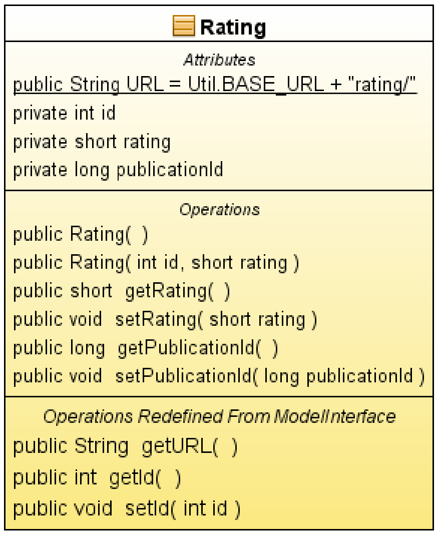
Publication

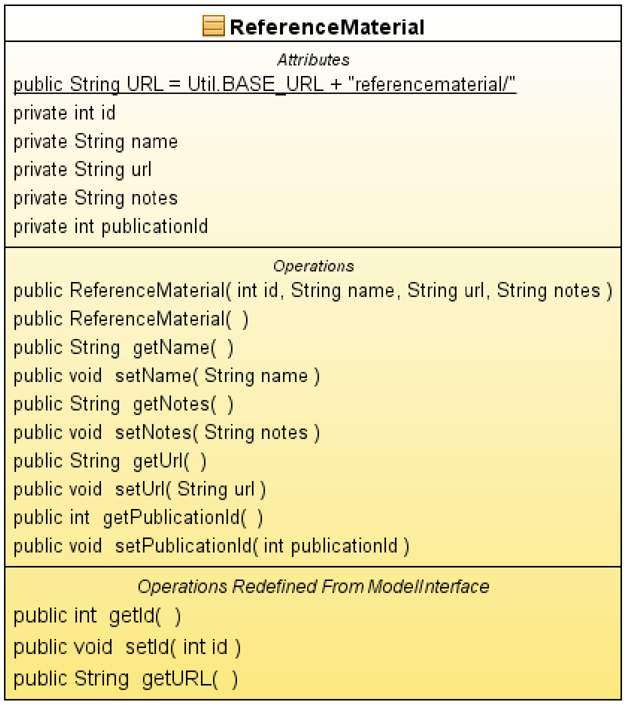
Attributes

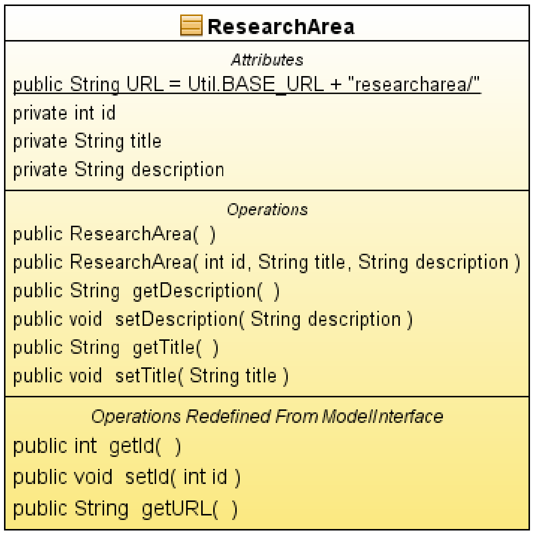
```
public String URL = Util.BASE_URL + "publication/"
private int id
private String address
private String booktitle
private String chapter
private String edition
private String editor
private String howPublished
private String institution
private String isbn
private String journal
private String number
private String organization
private String pages
private String publisher
private String series
private String publicationType
private String volume
private String title
private String month
private String note
private int year
private String abstracts
private int reviewStatus
private String doi
private boolean fullText
```

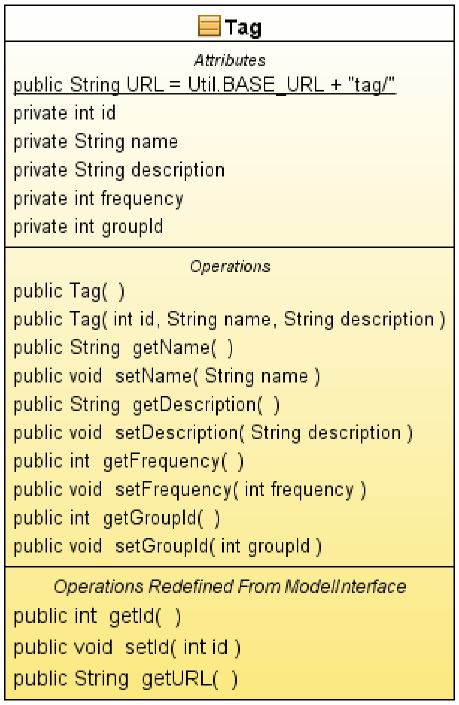
Operations Redefined From ModelInterface

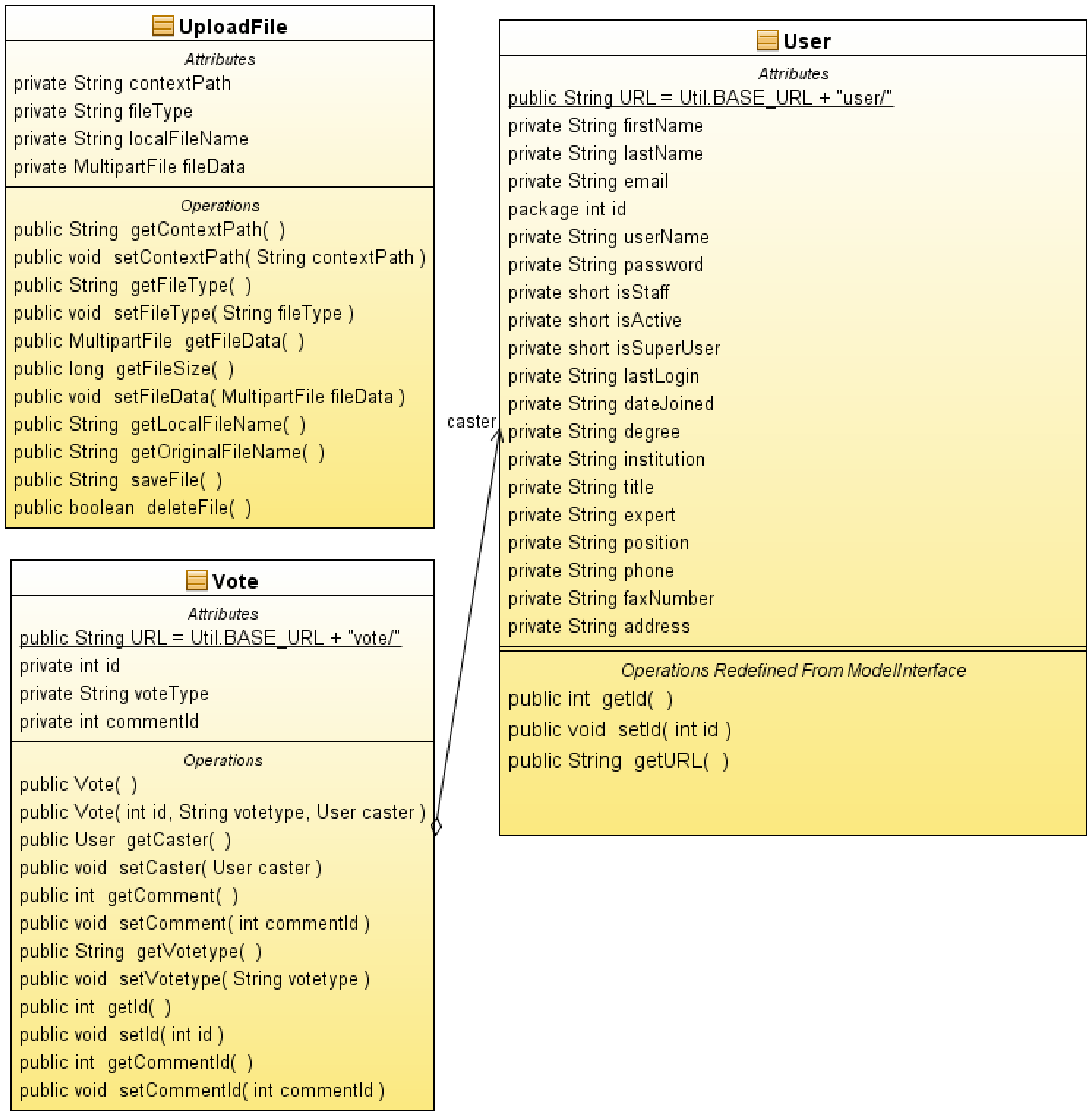
```
public int  getId(  )
public void  setId( int id )
public String  getURL(  )
```


Rating
Attributes
public String URL = Util.BASE_URL + "rating/"
private int id
private short rating
private long publicationId
Operations
public Rating( )
public Rating(int id, short rating)
public short getRating( )
public void setRating(short rating)
public long getPublicationId( )
public void setPublicationId(long publicationId)
Operations Redefined From ModelInterface
public String getURL( )
public int getId( )
public void setId(int id)


ReferenceMaterial
Attributes
public String URL = Util.BASE_URL + "referencematerial/"
private int id
private String name
private String url
private String notes
private int publicationId
Operations
public ReferenceMaterial(int id, String name, String url, String notes)
public ReferenceMaterial( )
public String getName( )
public void setName(String name)
public String getNotes( )
public void setNotes(String notes)
public String getUrl( )
public void setUrl(String url)
public int getPublicationId( )
public void setPublicationId(int publicationId)
Operations Redefined From ModelInterface
public int getId( )
public void setId(int id)
public String getURL( )


ResearchArea
Attributes
public String URL = Util.BASE_URL + "researcharea/"
private int id
private String title
private String description
Operations
public ResearchArea( )
public ResearchArea(int id, String title, String description)
public String getDescription( )
public void setDescription(String description)
public String getTitle( )
public void setTitle(String title)
Operations Redefined From ModelInterface
public int getId( )
public void setId(int id)
public String getURL( )


Tag
Attributes
public String URL = Util.BASE_URL + "tag/"
private int id
private String name
private String description
private int frequency
private int groupId
Operations
public Tag( )
public Tag(int id, String name, String description)
public String getName( )
public void setName(String name)
public String getDescription( )
public void setDescription(String description)
public int getFrequency( )
public void setFrequency(int frequency)
public int getGroupId( )
public void setGroupId(int groupId)
Operations Redefined From ModelInterface
public int getId( )
public void setId(int id)
public String getURL( )



```
UploadFile
Attributes
private String contextPath
private String fileType
private String localFileName
private MultipartFile fileData
Operations
public String  getContextPath( )
public void  setContextPath( String contextPath )
public String  getFileType( )
public void  setFileType( String fileType )
public MultipartFile  getFileData( )
public long  getFileSize( )
public void  setFileData( MultipartFile fileData )
public String  getLocalFileName( )
public String  getOriginalFileName( )
public String  saveFile( )
public boolean  deleteFile( )
```

```
Vote
Attributes
public String URL = Util.BASE_URL + "vote/"
private int id
private String voteType
private int commentId
Operations
public Vote( )
public Vote( int id, String votetype, User caster )
public User  getCaster( )
public void  setCaster( User caster )
public int  getComment( )
public void  setComment( int commentId )
public String  getVotetype( )
public void  setVotetype( String votetype )
public int  getId( )
public void  setId( int id )
public int  getCommentId( )
public void  setCommentId( int commentId )
```

```
User
Attributes
public String URL = Util.BASE_URL + "user/"
private String firstName
private String lastName
private String email
package int id
private String userName
private String password
private short isStaff
private short isActive
private short isSuperUser
private String lastLogin
private String dateJoined
private String degree
private String institution
private String title
private String expert
private String position
private String phone
private String faxNumber
private String address
Operations Redefined From ModelInterface
public int  getId( )
public void  setId( int id )
public String  getURL( )
```

Source Code

In this section sample code of controller, Validator, Comparator and service are documented.

1) FileUploadValidator.java(**Validation** Sample)

```
package frontend.model.validators;
import org.springframework.validation.Errors;
import org.springframework.validation.Validator;
import frontend.model.UploadFile;
 /**
 * This class validate the uploaded file by author
 * @author amir
 */
public class FileUploadValidator implements Validator{

          @Override
          public boolean supports(Class clazz) {
            //just validate the FileUpload instances
            return UploadFile.class.isAssignableFrom(clazz);
          }
          @Override
          public void validate(Object target, Errors errors) {
            UploadFile file = (UploadFile)target;
      //if file size is zero return an error code
            if(file.getFileData().getSize()==0){
                 errors.rejectValue("fileData", "required.fileUpload");
            }
          }}
```

2) Source code of publication controller(**Controller** Sample)

```
/**
 * This class is a controller for publication
 * all task about publication is done here
 * @author amir
 */
@Controller
@RequestMapping("/publication")
public class PublicationController {
    @Autowired
    PublicationServiceInterface publicationService;
    @Autowired
    UserServiceInterface userService;

    //An exception method for exception
    @ExceptionHandler(Exception.class)
    public ModelAndView handleException(Exception ex) {
        ModelAndView mav = new ModelAndView();
        mav.setViewName("error");
        mav.addObject("content", ex.getClass());
        return mav;
    }
    /**
     * return abstract of publication
     * @param publicationId
     * @param session
     * @param request
     * @return
     */
    @RequestMapping(value="/abstract/{publicationId}", method = RequestMethod.GET)
            public ModelAndView  getAbstract(@PathVariable("publicationId") String publicationId, HttpSession
session, HttpServletRequest request)
            {
        List<Tag> tagList=null;
        List<Tag> thisPublicationTagsList=new ArrayList<Tag>();
        String loginStatus=userService.getLastLogInStatus(session);
        //calculate the tag's weight
        if(Util.isLocalData(session))
          tagList=PublicationService.getTagCloud(SampleData.getLocalData(session).getPublicationList());
        else
          tagList=PublicationService.getTagCloud(publicationService.fetchPublicationWithDetails(null, session));
        User user=(User)session.getAttribute(Util.USER_SESSION);
        Login login=new Login("username", "password","/publication/abstract/".concat(publicationId),loginStatus);
        ModelAndView mav = new ModelAndView();
        mav.setViewName("publication_abstract");
        Publication publication= publicationService.getPublicationById(Integer.parseInt(publicationId),session);


        for(Tag pubtag: publication.getTags()){
          for(Tag tagcloud: tagList){
            if(pubtag.getId()==tagcloud.getId()){
              thisPublicationTagsList.add(tagcloud);
              pubtag.setFrequency(tagcloud.getFrequency());


            }
          }
        }


        mav.addObject("publication",publication);
        mav.addObject("login",login);
        mav.addObject("user", user);
        mav.addObject("tagCloud",thisPublicationTagsList);
        if(session.getAttribute("rating")!=null)
            mav.addObject("rating", ((String)session.getAttribute("rating")));
        else
            mav.addObject("rating","");
        return mav;
                    }
    /**
     * add new tag to publication
     * @param session
     * @param tagName
     * @param tagDescription
     * @param backAddress
     * @param publicationId
     * @return
     */
    @RequestMapping(value="/tag/add", method = RequestMethod.POST)
    public ModelAndView addTag(HttpSession session, @RequestParam("tagName") final String tagName,
    @RequestParam("tagDescription") final String tagDescription,
    @RequestParam("backAddress") final String backAddress,
    @RequestParam("publicationId") final int publicationId) {
        if(userService.isLogIn(session)){
            if(!Util.isLocalData(session))
                publicationService.addTag(session, tagName, tagDescription, publicationId);
            else
                SampleData.getLocalData(session).addTag(tagName, tagDescription, publicationId);
        }
        return new ModelAndView("redirect:"+backAddress);
```

```
}
/**
 * it is called when user login
 * @param session
 * @param backAddress
 * @param login
 * @return
 */
@RequestMapping(value="/login", method = RequestMethod.POST)
public ModelAndView getPage(HttpSession session, @RequestParam("backAddress") final String backAddress,
@ModelAttribute("login") Login login) {
    userService.login(login,session);
    return new ModelAndView("redirect:"+backAddress);
}
/**
 * called when logout
 * @param session
 * @param backAddress
 * @return
 */
@RequestMapping(value="/logout", method = RequestMethod.GET)
public ModelAndView logout(HttpSession session, @RequestParam("backAddress") final String backAddress) {
    userService.logout(session);
    return new ModelAndView("redirect:"+backAddress);
}
/**
 * add comment to the publication
 * @param session
 * @param comment
 * @param backAddress
 * @param publicationId
 * @return
 */
@RequestMapping(value="/comment/add", method = RequestMethod.POST)
public ModelAndView addComment(HttpSession session, @RequestParam("comment") String comment,
@RequestParam("backAddress") final String backAddress,
@RequestParam("publicationId") final int publicationId) {
    User user=(User)session.getAttribute(Util.USER_SESSION);
    if(userService.isLogIn(session)){
        if(!Util.isLocalData(session)){
            publicationService.addComment(session, comment, publicationId, user);
        }
        else
            SampleData.getLocalData(session).addComment(comment, publicationId, user);
    }
    return new ModelAndView("redirect:"+backAddress);
}
/**
 * delete comment from publication
 * user must be owner of comment
 * @param session
 * @param id
 * @param backAddress
 * @param publicationId
 * @return
 */
@RequestMapping(value="/comment/delete", method = RequestMethod.GET)
public ModelAndView deleteComment(HttpSession session, @RequestParam("id") final int id,
@RequestParam("backAddress") final String backAddress, @RequestParam("publicationId") final int publicationId) {
    User user=(User)session.getAttribute(Util.USER_SESSION);
```

```
        if(userService.isLogIn(session)){
            if(!Util.isLocalData(session)){
                publicationService.deleteComment(session, id, publicationId, user);
            }
            else
                SampleData.getLocalData(session).deleteComment(id, user, publicationId);
        }
        return new ModelAndView("redirect:"+backAddress);
    }
    /**
     * add rating to the publication
     * @param session
     * @param rate
     * @param backAddress
     * @param publicationId
     * @return
     */
    @RequestMapping(value="/rating/add", method = RequestMethod.POST)
    public ModelAndView addRating(HttpSession session, @RequestParam("group") final int rate,
    @RequestParam("backAddress") final String backAddress, @RequestParam("publicationId") final int publicationId) {


            if(!Util.isLocalData(session))
                publicationService.addRate(session, rate, publicationId);
            else
                SampleData.getLocalData(session).addRating(rate, publicationId);
            session.setAttribute("rating",Integer.toString(rate));
        return new ModelAndView("redirect:"+backAddress);
    }


}
```

3) UserService .java (**Service** sample)

```
/**
 * Do login, logout, session tasks
 * @author amir
 */
@Service
public class UserService implements UserServiceInterface{
   @Autowired
   private RestTemplate restTemplate;
   @Autowired
   WebService<User> userWebService;
   @Override
   public User getUser(String url, HttpSession session) {
      return userWebService.get(User.class, url, session, null);
   }
/**
    * return true if user is loggin
    * @param session
    * @return
    */
   @Override
   public boolean isLogIn(HttpSession session) {
      if(session.getAttribute(Util.USER_SESSION_ID)==null ||
((String)session.getAttribute(Util.USER_SESSION_ID)).isEmpty()
||((User)session.getAttribute(Util.USER_SESSION))==null){
         logout(session);
         return false;
      }
      else
```

```
            return true;
    }
/**
     * return appropriate string message for user status
     * @param session
     * @return
     */
    @Override
    public String getLastLogInStatus(HttpSession session) {

if(session.getAttribute(Util.LAST_LOGIN_STATUS)==null||((String)session.getAttribute(Util.LAST_LOGIN_STATUS)).i
sEmpty())
            return Util.LOGIN_STATUS_NO_TRY;
        else{
            String result= ((String)session.getAttribute(Util.LAST_LOGIN_STATUS));


            if(result.equals(Util.LOGIN_STATUS_FAILD)){
                session.setAttribute(Util.LAST_LOGIN_STATUS, Util.LOGIN_STATUS_NO_TRY);
                return Util.LOGIN_STATUS_FAILD;
            }
            return Util.LOGIN_STATUS_SUCCESSFUL;
        }
    }
    /**
     * if login be successful return true
     * @param login
     * @param session
     * @return
     */
    @Override
    public boolean login(Login login, HttpSession session) {
```

```
if(!Util.isLocalData(session)){
    HttpEntity<Login> entity  = new HttpEntity<Login>(login);
    ResponseEntity<Login> response=null;
    try{
        response = restTemplate.postForEntity(Login.URL, entity, Login.class);
        Login result = response.getBody();
        HttpHeaders headers = response.getHeaders();
        String sessionId=Util.getHeaderValue("set-cookie",headers);
        if(sessionId==null||sessionId.trim().isEmpty()) {
            session.setAttribute(Util.LAST_LOGIN_STATUS, Util.LOGIN_STATUS_FAILD);
            return false;
        }
        sessionId=sessionId.substring(0,sessionId.indexOf(";"));
        session.setAttribute(Util.USER_SESSION_ID, sessionId);
        session.setAttribute(Util.LAST_LOGIN_STATUS, Util.LOGIN_STATUS_SUCCESSFUL);
        login.setUserAtom(result.getUserAtom());
        User user = userWebService.get(User.class, login.getUserAtom().getHref(), session, null);
        login.setUser(user);
        session.setAttribute(Util.USER_SESSION,user);
        return true;
    }
    catch(RestClientException ex){
        session.setAttribute(Util.LAST_LOGIN_STATUS, Util.LOGIN_STATUS_FAILD);
        System.out.println("RestClientException in Login:"+ex.getMessage());
        if(ex instanceof HttpServerErrorException){
            HttpServerErrorException httpServerErrorException=(HttpServerErrorException)ex;
            throw new IllegalStateException(httpServerErrorException.getResponseBodyAsString());
        }
        return false;
    }
```

```
            catch(Exception ex){
                session.setAttribute(Util.LAST_LOGIN_STATUS, Util.LOGIN_STATUS_FAILD);
                System.out.println("Rest Call Exception in Login:"+ex.getMessage());
                return false;
            }
        }
        else{
            return SampleData.getLocalData(session).login(login, session);
        }
    }
/**
     * logout from system
     * @param session
     */
    @Override
    public void logout(HttpSession session) {
        if(true || isLogIn(session)){
            session.setAttribute(Util.USER_SESSION,null);
            session.setAttribute(Util.USER_SESSION_ID,"");
            session.setAttribute(Util.LAST_LOGIN_STATUS,Util.LOGIN_STATUS_NO_TRY);
        }
    }

}
```

4) PublicationRateComparator .java (**Comparator** sample)

```
/**
 *compare publication based on their rating value
 * @author amir
 */
public class PublicationRateComparator implements Comparator<Publication>{
    //descending sort
    @Override
    public int compare(Publication pub1, Publication pub2) {
        int result=(pub1.getAverageOfRates()>pub2.getAverageOfRates()? -1 :
(pub1.getAverageOfRates()<pub2.getAverageOfRates()? 1 : 0));
        if(result!=0) return result;
       result=(pub1.getCountOfRates()>pub2.getCountOfRates()? -1 : (pub1.getCountOfRates()<pub2.getCountOfRates()? 1 : 0));
        return result;
    }
}
```

Appendix B

Questionnaire Form

Knowledge Platform's Front-End

Evaluation

1) What is your last qualification?
    - PhD
    - M.Sc.
    - B.Sc.

2) Is your job related to computer?
    - Yes
    - No

3) How much are you familiar with peer review process?
    - Excellent
    - Above Average
    - Average
    - Below Average
    - Very Poor

4) Do you usually use scientific literature and digital library (Such as CiteSeer)?
    - Very Frequently
    - Frequently
    - Occasionally
    - Rarely
    - Very Rarely
    - Never

5) Front-End provides a good mechanism for categorization of content by users (Such as tagging and tag cloud features)
    - Strongly Agree
    - Agree
    - Disagree
    - Strongly Disagree

6) Front-End provides a good mechanism for finding information through keyword and tag searching.
    - Strongly Agree
    - Agree
    - Disagree
    - Strongly Disagree

7) Front-End provides a user friendly environment for exchanging views and collaborating. (comment, rating, vote)
    - Strongly Agree
    - Agree
    - Disagree
    - Strongly Disagree

8) Front-End support *easy-to-use features for* linking and referencing of information. (Such as using hyperlink for finding more info about authors, references, keywords, citation)
    - Strongly Agree
    - Agree
    - Disagree
    - Strongly Disagree

9) Front-End provides a good mechanism for authors to submit manuscripts online. (Upload manuscripts/supplementary files, Meta data such as title, abstract, authors)
    - Strongly Agree
    - Agree
    - Disagree
    - Strongly Disagree

10) Front-End supports a good tracking system for authors during publishing process.
    - Strongly Agree
    - Agree
    - Disagree
    - Strongly Disagree

11) Front-End provides a good Editorial board mechanism. (assigning an article to reviewer, monitor manuscript status)
    - Strongly Agree
    - Agree
    - Disagree
    - Strongly Disagree

12) Front-End supports a good Reviewer tracking system.(Such as notification, response template)
    - Strongly Agree
    - Agree
    - Disagree
    - Strongly Disagree