



UNIVERSITA' DEGLI STUDI DI TRENTO

Facoltà di Scienze Matematiche, Fisiche e Naturali



UNIVERSITY OF TRENTO - Italy

Corso di Laurea Specialistica in Informatica
within European Master in Informatics

Final Thesis

A Service-Oriented Architecture for the NEEDLE (Next
gEneration sEarch engine for Digital LibrariEs) Multimodal
Search Engine

Relatore/1st Reader:

Prof. Marco Ronchetti
University of Trento, Italy

Laureando / Graduant:

Muthiyalu Jothir Navaneetha Krishnan

Contro Relatore/ 2nd Reader:

Prof. Dr. Matthias Jarke
RWTH Aachen, Germany

Anno Accademico 2006 - 2007

Acknowledgment

I would like to express my sincere thanks to Prof. Marco Ronchetti, Department of Information and Communication Technology, University of Trento, for being my advisor for the thesis work. He has been a great support and guidance right from the proposal of the topic till the completion of the work. The valuable discussions, key ideas and continuous feedback at every stage greatly helped me to learn and explore new technologies. The moral encouragement provided by him helped me to deliver the best. Thank you Sir.

I would also like to thank Prof. Dr. Matthias Jarke, Informatik 5, RWTH, Aachen, for kindly accepting to be my thesis advisor at RWTH, Aachen. Thank You Sir.

It has been a great learning experience to work with Ms. Angela Fogarolli, PhD student at University of Trento. She helped me to get familiar with the application and provided valuable feedback at every stage of the work. She has been a great source of encouragement.

I would be failing in my duty if I miss to thank my parents, friends, colleagues and all those who have directly or indirectly helped me in the thesis work. Thank you all for the support and encouragement.

Table of Contents

Acknowledgment	2
Abstract	5
1 Introduction	7
1.1 E-Learning	7
1.2 NEEDLE	8
1.3 Multimedia Search	8
2 Thesis – Goal, Scope & Approach	11
2.1 Goal	11
2.2 Scope	13
2.3 Approach	14
3 NEEDLE & Other E-Learning Applications – Comparative Study	16
3.1 NEEDLE	16
3.1.1 Overview	16
3.1.2 System Architecture	17
3.2 E-Learning Applications – Literature study	18
3.2.1 ATLAS - Architecture for Transcription, Localization, and Transcription Engineering System	18
3.2.2 E-Learning Framework	20
3.2.3 Search Service for Heterogeneous e-Learning Resources	21
3.2.4 AdELE - Adaptive E-Learning through Eye Tracking	22
4 NEEDLE Re-engineering using J2EE Design Patterns & Hibernate	25
4.1 NEEDLE: Prototype	25
4.1.1 Need for Re-engineering	25
4.2 Java 2 Enterprise Edition – J2EE	26
4.2.1 Evolution	26
4.2.2 Introduction to J2EE	27
4.2.3 Components and Containers	27
4.2.4 Presentation Tier: Web Components	29
4.2.5 Business Tier: Enterprise JavaBeans	31
4.3 Design Patterns	34
4.3.1 J2EE Patterns	34
4.3.2 Model View Controller (MVC) Pattern	35
4.3.3 MVC Pattern in NEEDLE Prototype	36
4.4 Data Access Layer	38
4.4.1 Java Database Connectivity (JDBC)	38
4.4.2 Hibernate	39
4.4.3 Hibernate Implementation in NEEDLE	41
5 NEEDLE Automation Tool	43
5.1 Objective	43
5.2 Scope	44
5.3 Requirements	44
5.4 Design	45
5.4.1 Sequence Diagram for Needle Upload tool	46
5.5 Implementation	48
5.5.1 Lucene API	49
5.5.2 QuickTime API	52

5.5.3	MP4BOX	54
5.5.4	NEEDLE Implementation.....	55
6	NEEDLE Advanced Search Interface.....	61
6.1	Overview.....	61
6.2	Advanced Search Techniques	61
6.3	Advanced Search Techniques in NEEDLE	64
6.4	Ajax.....	67
6.4.1	Overview	67
6.4.2	Classical Web Interaction Model.....	67
6.4.3	Ajax Interaction Model.....	68
6.4.4	JavaScript to Ajax Technology.....	69
6.4.5	Scenarios showing Ajax interactions	70
7	Service Oriented Architecture for NEEDLE Search Library.....	73
7.1	Service Oriented Architecture - Evolution	73
7.2	Web Service Architecture (WSA)	74
7.3	Interoperability Architecture.....	74
7.4	SOA Infrastructure.....	76
7.4.1	SOAP	77
7.4.2	WSDL	81
7.4.3	UDDI (Universal Description, Discovery, and Integration).....	83
7.5	Apache Axis2.....	85
7.5.1	Creating Web Services Using Axis2.....	85
7.5.2	Generating Service code from WSDL	87
7.5.3	Axis2 code generator Plug-in.....	89
7.6	NEEDLE Search Library	94
7.6.1	Motivation.....	94
7.6.2	Requirements	95
7.6.3	Design	96
7.6.4	Presentation Tier	97
7.6.5	Search Library Configuration File – XML File.....	97
7.6.6	Search Client.....	99
7.6.7	Web Service	99
7.6.8	Implementation	100
7.7	Search Library Customization Tool.....	102
7.7.1	Motivation.....	102
7.7.2	Requirements	102
7.7.3	Detailed Screen Description	103
8	Future Work	107
9	Conclusion	110
	APPENDIX.....	111
	Bibliography	115

Abstract

There has been a great amount of research carried over in the development of e-Learning technologies with particular importance on web-based applications. In this thesis, we discuss the work on the design and implementation of one such modern, evolving e-Learning application. The main objective of the thesis is to analyze the requirements, re-engineer, design and implement a web based prototype for NEEDLE (Next gEneration sEarch engine for Digital LibrariEs) system. NEEDLE is an e-learning application which aims at indexing, searching and presenting structured and unstructured multimedia data. Though the existing prototype had the essential search functionalities, the system had to be re-engineered to support advanced search functionalities and to build a prototype based on Service Oriented Architecture (SOA). The research methodology consisted of extensive literature study of existing e-learning applications and a comparison of design/architecture approach. Also, the best practices of Java/J2EE design patterns have been incorporated in the prototype. State of the Art Technologies like Web Services, Hibernate, Ajax, Lucene Search API, PowerML API, QuickTime API etc. have been used to enhance the functionalities. We introduce a new concept called “*Search library*” which is developed as independent web service capable of performing search with external search engines. It could be plugged into the prototype to extend the search domain of NEEDLE. These pluggable components offer the flexibility to transform the NEEDLE system into a Service Oriented Architecture. The new prototype has been thoroughly tested and evaluated. The thesis concludes with a discussion on the potential future work that could be carried over in the future.

Chapter 1:

Introduction

1 Introduction

1.1 E-Learning

The recent years have seen a phenomenal growth in the use of Internet in student education, with various reasons encouraging the implementation of Web based technology. This technological development has the strength to change and restructure the traditional models of education (class room teaching), mainly the delivery and interaction in and with course materials and associated resources. Actually it is changing the way universities teach and students learn.

Now it happens to be the case where the students type their notes on computers and professors send them the lecture notes and other work assignments by e-mail. The drive has been a shift from teacher centered education to learner-centered education, encouraging the educators to provide courses which facilitate students to handle their own learning. Simply defined, e-Learning offers the means to deliver courses to new and different audiences who may be dispersed across the globe and who may not have the opportunity to study in a direct University or school environment.

E-learning is not only used in University / School education, but also widely used in corporate training courses. Apart from few basic classroom training, majority of the knowledge sharing takes place through e-learning. New entrants learn business, technology, soft skills etc. by using such systems. These systems need careful design to provide the best user interactivity and flexibility in learning.

Thus, to attract learners towards an e-learning system, the only way is to provide the best features and resources to them. It would be best if the e-learning system could be personalized according to the standard and preference of the learner. Researchers are working on a wide variety of challenges yet to be solved in the e-learning arena. Recently, there has been an increased usage of multimedia data in almost all the fields. Students expect more than just notes and documents. The teachers have also well understood the situation and are coming up with interactive media and attractive means of delivering lectures. Distribution of video / audio lectures [4 & 5] has become quite common.

Web based technologies and distributed computing architecture have evolved out quite successfully to support the sharing and distribution of such huge volumes of multimedia data. Buzz words like Service Oriented Architecture (SOA), Semantic web, Web mining have started to be an integral part of modern e-Learning applications. Thus, it is necessary to carefully design a system by proper usage of such advanced technologies offered by the research world.

In this work, we are going to study one such e-Learning system called NEEDLE – Next gEneration sEarch engine for Digital LibrariEs.

1.2 NEEDLE

NEEDLE – Next gEneration sEarch engine for Digital LibrariEs [1] - is an e-learning application which aims at indexing, searching and presenting structured and unstructured multimedia data. The system provides a way to search e-learning materials through a web-based search interface.

The e-Learning materials consist of video lectures and corresponding audio tracks, PowerPoint presentation slides etc. The application's main objective is to present the structured and unstructured multimedia e-Learning materials. The users could query the NEEDLE system to search for materials of their interest.

The data i.e. video lectures and slides, for NEEDLE come from the LODE system. LODE is web-based application for presenting the video lectures synchronized with presentation slides [2 & 3]. We discuss more about LODE in the next chapter. The audio content of the video lectures from the LODE system is transcribed using speech recognition and speech processing tools. The text content of the transcription and PowerPoint slides are indexed and searched using NEEDLE system.

In this study, we are going to analyze and re-engineer the NEEDLE system in order to deliver additional functionalities. The main goal here is to provide a better, user-friendly search interface. Since NEEDLE is a multimedia search interface, it would be better to see a brief introduction of such applications.

1.3 Multimedia Search

The recent improvements in digital multimedia and Web technologies have resulted in the wide usage and distribution of several media types like images, audio, video, text etc. in web pages. In such a scenario, it is essential for users to perform searches on these multimedia data.

Most of the commercial search engines only offer text based search and few also provide image search. However, there is still a need for searching video, audio, graphics etc. Commercial video hosting sites like YouTube that offer search for video actually performs the search only on the meta-data (text content describing the video) attached with the video. They do not search the video / audio content. To perform

such advanced searches additional tools and technologies are required. Also efficient algorithms are needed for searching images, video/ audio content.

Researchers are working on providing a multimedia web search engine that accepts a query and searches not only text contents, but also images, video, audio for the occurrence of keywords and provides a categorized output to the user indicating the type of media. For e.g. the keyword “Ferrari” could be found in a sports article (web page), in a picture with embedded words as “Ferrari”, in a video collection of a F1 race etc.

The design of interface for searching multimedia content is also quite tricky. The user could have an option of selecting only particular media types to search. The input given by the user consists of keywords or phrases called query. However, in the near future search engines could accept images or video sample or audio tracks as input, perform search and find similar images etc. These ideas take us to technologies like pattern recognition, pattern matching etc. All these notions are considered while designing a multimedia search interface. The search results presented to the user usually indicates the type of media and could have options to open and play the file e.g. in a video player etc.

In our case, NEEDLE is a multimedia search engine which performs search on video / audio lectures and PowerPoint slides. The occurrence of the keyword in the video lecture is marked with a timestamp and presented as search result. Hence, when the user clicks the video link, it plays from the specific timestamp. Similarly, the text in the PowerPoint slides is searched and the particular slide where the keyword occurs is noted and presented as image to the user.

Outline of the Report

In the next chapter, we discuss the main objective and scope of this thesis work. Next, we proceed with the details of design and implementation of NEEDLE prototype. In chapters 4 to 6, we analyze the ways to improve the search interface and we introduce advanced search functionalities in the NEEDLE prototype. In the final chapters, we move our discussion to the new paradigm of Service Oriented Architecture (SOA) and how it could be used to improve NEEDLE system. In order to enable NEEDLE to interact with external search engines and e-Learning resources, we introduce the concept of Search libraries which are implemented using web services (Service Oriented Architecture). Finally, we conclude the report with a discussion on potential future work that could be carried over.

Chapter 2:

Thesis –

Goal, Scope & Approach

2 Thesis – Goal, Scope & Approach

2.1 Goal

The following are the main goals of the thesis work with NEEDLE,

1. Enhancement of NEEDLE prototype to provide advanced search functionalities, efficient and user – friendly searching and to follow better design patterns.
2. Provide an open architecture for NEEDLE to interact with external applications and expand the search domain.
3. Provide additional utility tools for automation of some key procedures required for functioning of NEEDLE.

We will discuss about each of these goals in the next paragraphs.

1. Enhancement of NEEDLE prototype

There has been a great amount of research and development carried out in the development of the NEEDLE prototype. The web-based application built on enterprise architecture has simple and efficient features to perform search on the e-Learning materials. Just as in any search engine, the user provides a query (keyword or phrase) to be searched. The system responds with a list of e-Learning resources with links to the video lectures, slides etc.

Though this search interface is simple and efficient, it would be better to provide an advanced search screen with more options to help the user to refine the query and narrow down the search results.

Also, the prototype which is built using Java 2 Enterprise Edition (J2EE) architecture needs to be re-engineered to follow better design patterns and more efficient technologies to provide maintainability and proper separation of code. One such task would be to re-implement the data access layer with Hibernate technology replacing the existing Java Database Connectivity (JDBC). The design pattern used in the presentation-tier and business-tier is roughly the Model-View-Controller (MVC) pattern. However, it needs to be re-designed to provide a clear separation of presentation- tier from business logic.

2. Open Architecture for NEEDLE

NEEDLE contains e-Learning materials received from the LODE system which are pre-processed made available for searching. The simple search features in the existing system and the advanced search functionalities to be introduced would help the users to refine the queries and retrieve the desired results. However, from a user's perspective, the resources are limited in using just NEEDLE alone. It would be better to provide a broader domain of searching for the materials.

NEEDLE should be able to collaborate with external e-Learning applications and other general search engines to search for e-Learning materials. Also, other applications could take advantage of NEEDLE resources and have a mutual benefit.

From literature study [8], we understand that many e-Learning frameworks are based on Service Oriented Architecture (SOA) to allow other applications to interact with them and vice versa. Hence, we study the possibilities of implementing SOA in NEEDLE and thus extending the search domain for e-Learning materials. We introduce a concept called "*Search Library*" implemented as independent web services which act as a bridge between NEEDLE and external search engines.

3. Utility Tools for NEEDLE

The e-Learning materials from LODE system have to be pre-processed and corresponding database entries needs to be updated to make them searchable. There are sequences of complicated steps to be executed to achieve this process, which are error prone and previously done manually. In order to automate this process, it is required to design and implement a tool.

There is also a need for a tool to assist developers to customize a new "*Search Library*" and integrate with NEEDLE system. This tool works like a wizard to configure web service specific properties which would be stored in an XML file and used by NEEDLE business logic to invoke the web service.

2.2 Scope

The thesis work deals with an evolving, web based e-Learning application which aims at presenting structured and unstructured multimedia data. Hence, throughout this work we discuss about various challenges related to web application architectures, search interfaces, multimedia information retrieval etc. and also the solutions provided by the state-of-the-art-technologies.

This work will be of interest to researchers and students working on e-Learning applications. We also discuss some of the ideas from other existing e-Learning frameworks and related research work. This would help to understand the key features, design architectures and implementation technologies used in various systems.

As a major portion of the thesis deals with practical implementation and prototype enhancement, there will be good chance to learn and experiment some of the latest technologies and programming APIs. To name a few of them, J2EE Design patterns, Hibernate, Lucene API, Ajax, QuickTime for Java API, Web Services – SOAP, WSDL, Apache Axis2 etc.

With an objective to introduce an open architecture for NEEDLE, we introduce Service Oriented Architecture (SOA) and discuss in detail about the key components of the technology and the implementation details.

Therefore, the thesis provides scope for learning latest state-of-the-art-technologies and to develop ideas to implement next generation open architecture systems.

2.3 Approach

The first step was to conduct a thorough study of NEEDLE and understand the implementation details of the existing prototype. In parallel, similar e-Learning applications were studied to collect ideas about their design and implementation. This sort of comparative study provides an efficient way of analyzing the drawbacks of the existing system and throws light on areas to be improved.

Once the architecture and design issues were identified, it was necessary to introduce new functionalities and re-engineer others. A wide range of technologies had to be studied to select the best fit for the particular task. The technology need not be complex or the most sophisticated. Rather, it should be able to suit the requirements in a simple, developer-friendly manner and also should provide scope for future extension. Open source libraries and APIs were identified and used in the implementation.

The first implementation task was to re-engineer and enhance the NEEDLE prototype. The web-based prototype was re-designed using standard design patterns and with major changes to the data access layer. New search functionalities were implemented. The utility tools were also developed to automate some of the core procedures which were previously done manually.

Finally, the Service Oriented approach was studied and implemented in NEEDLE to provide an open architecture and to enable interaction with external search engines and e-Learning repositories.

Throughout the development process, there has been a parallel theoretical study along with practical implementation. The proper mix helped a lot to understand the technologies and provide the best output.

Chapter 3:

NEEDLE &

Other E-Learning Applications –

Comparative Study

3 NEEDLE & Other E-Learning Applications – Comparative Study

3.1 NEEDLE

3.1.1 Overview

NEEDLE – Next gEneration sEarch engine for Digital LibrariEs [1] - is an e-learning application developed as a research work carried out at Department of Information & Communication Technology, University of Trento, Italy. It aims at indexing, searching and presenting structured and unstructured multimedia data [1]. The system provides a way to search e-learning materials through a web-based search interface.

The research team has a huge collection of video lectures and seminars ranging more than 400 hours. The corresponding PowerPoint slides shown during the lecture are synchronized with the video. These combined materials are available from LODE [2 & 3], which is an application used for the acquisition of video lectures along with the corresponding PowerPoint slides and presents it on a web-based prototype with tools to assist the user to navigate through the lecture.

The need was to build an efficient search interface for searching this huge collection of multimedia data (video lectures and slides) stored with related metadata like timestamps etc. Speech recognition and Speech processing tools were used to transcribe the audio content of the video lecture and stored in a transcript file. NEEDLE uses the text content of these audio transcripts and slides to perform the search.

In order to search the multimedia content from the LODE system, the data needs to be pre-processed and some database entries needs to be inserted. Specifically, the text content is indexed using Lucene API and the created indexes are stored in specific directory structures. The information about the directory structures is stored in database tables.

The search results are presented with links to directly play the video where the search keyword occurs and also displays the corresponding slides as images.

The NEEDLE web-based prototype has been built using J2EE technology with JSP for presentation, Servlet and JavaBeans for business logic and Java Database Connectivity (JDBC) for data access.

In the next section, we discuss the system architecture of NEEDLE and then proceed with a literature study of similar e-Learning applications.

3.1.2 System Architecture

To have a better understanding of the relationship between LODE and NEEDLE, we discuss the System Architecture. Fig 3.1 shows the System Architecture of NEEDLE taken from [1].

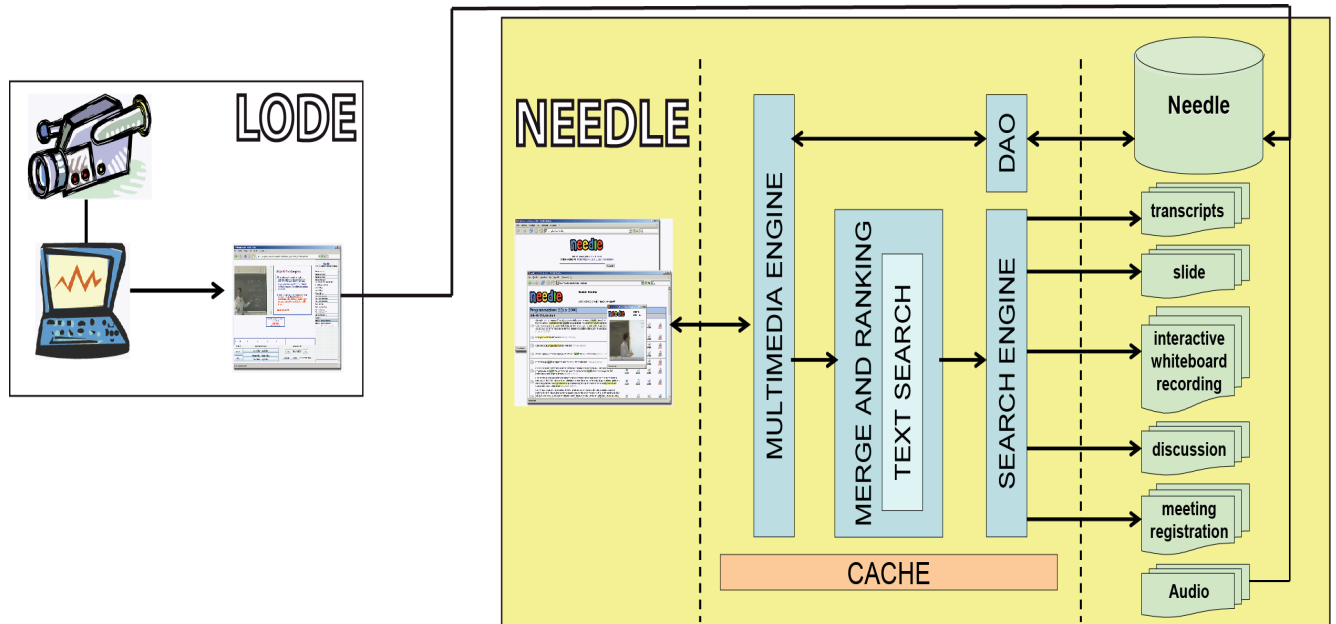


Fig. 3.1 NEEDLE System Architecture

LODE provides the complete backend data for NEEDLE. The captured video lectures with synchronized PowerPoint slides, audio transcripts etc. are fed into the NEEDLE backend.

Search Engine – This component uses Lucene API to search the indexed text data.

Merge and Ranking – This component is responsible for integrating the search results and presenting to the user. The search keywords might occur only in a video lecture or only in a presentation slide or maybe in both.

Multimedia Engine - This takes care of playing the video / audio content in a separate video player (QuickTime Player) and display of the slides as images.

3.2 E-Learning Applications – Literature study

In order to understand, design and re-engineer the NEEDLE application, it is necessary to conduct a literature study and collect ideas from various existing e-Learning applications and related research work. We briefly discuss the features of few of these applications in this section. Some of them consist of implemented prototypes and some other provides framework specification for development of e-Learning applications.

In this section, we discuss four different e-Learning applications. Two of them show the importance of Service Oriented Architecture in e-Learning environments and prescribe a framework. Another application deals with the management of multimedia resources with semantic annotation technologies. Finally, we see an interesting application which mainly focuses on personalization and adaptivity in e-Learning systems.

3.2.1 ATLAS - Architecture for Transcription, Localization, and Transcription Engineering System

ATLAS [6] is a web based e-Learning architecture which aims at management of multimedia artifacts in virtual communities. The key motivation of the work is to provide an efficient mechanism for distribution and management of the multimedia e-Learning resources created within the community of collaborating scientists. The main focus of the work is on humanist scientist communities which are observed to have more discursive nature and wider creation and usage of multimedia resources.

The work initially discusses the various strategies in the management of multimedia content in e-Learning communities. It has been found that adding semantic annotation or semantic meta-data to the multimedia artifacts helps in better classification, organization and management. In order to provide semantic annotations and structuring, Ontologies-based Information systems were introduced. However, Ontologies suffer from lack of flexibility and has to be suitable for a wide range of user interpretations in the community. To overcome these disadvantages, domain specific Ontologies are created after careful analysis and in an incremental fashion. As a final result, an Ontology containing the terminology used in the domain is required to manage the multimedia data.

The following three concepts are discussed relating to Ontology-based multimedia content management,

- ✓ Transcription – This aims at converting the multimedia content into a readable format for the users. E.g. Annotation of the multimedia content and converting into XML format.

- ✓ Localization – Creation of domain specific ontology for structuring or classifying the multimedia content.
- ✓ Re – addressing – Process of presenting the semantic-structured multimedia content back to the users.

The research work uses a multimedia meta-data standard called MPEG 7, which is an XML-based multimedia content description standard introduced by moving pictures experts group (MPEG). MPEG 7 is used to build the ontology and achieves transcription, localization and re-addressing.

Finally, the work discusses the architecture of the ATLAS application which is based on MPEG 7 Ontology. The ATLAS architecture works with a community repository which stores the created multimedia content and a MPEG 7 XML repository containing the Ontology. The stored content could be browsed, searched and personalized based on meta-data and the semantic information stored in the Ontologies. The system discusses the methods to tackle storage issues of the huge amount of data and issues relating to network transmission of movies. The multimedia content is presented along with metadata on a Java based MPEG 7 player.

Comparison with NEEDLE

The scope of ATLAS software framework is completely different from that of NEEDLE. ATLAS is meant for multimedia materials management in an e-Learning community whereas NEEDLE aims at presenting multimedia resources through a web based user interface and does not involve in community activity.

Although, the basic purpose of the two systems is different, both aim at managing, structuring and presenting multimedia content. The semantic annotation of multimedia data used in ATLAS achieved using Ontologies helps it to structure and manage the content within the particular domain / community. The current research work on NEEDLE also aims at information extraction from multimedia e-Learning materials and developing Ontologies to classify the resources. This would enable searching based on semantics in addition to mere text-base search.

3.2.2 E-Learning Framework

Joint Information Systems Committee (JISC) has carried out research work and has identified the need for a common, standardized framework specification for developing e-Learning application. They have introduced the E-learning Framework (ELF) [8] in collaboration with other research institutes to support interoperable e-Learning applications.

ELF imposes a Service Oriented Architecture (SOA) to be followed to implement two types of services, namely: e-Learning domain specific services and common services.

The following are some e-learning domain services

- ✓ *Course management* – Service used to create, manage and organize the course materials.
- ✓ *Student assessment / Grading* – Service used to assign marks/grades for students with respect to a particular course etc.
- ✓ *Curriculum* – Service used to manage the curriculum of a particular e-Learning course.
- ✓ Etc.

The following are some common services,

- ✓ *Authentication* – Service used to identify and authenticate the users of an application.
- ✓ *Content Management* – Service used to store, organize, classify, retrieve the huge volumes of e-Learning materials created and distributed across applications. Specifically, multimedia content management is of more importance in e-Learning environments.
- ✓ *Digital Rights Management* – Service takes care of copyright policies etc. related to the e-Learning resources.
- ✓ *Federated Search* – Service which provides a common search interface to discover resources across multiple, heterogeneous repositories. It uses search protocols like SRW, XQuery, Z39.50 etc. It differs from an ordinary Search service which just performs search on a single repository.

In the next section, we see a research work that implements a federated search service.

3.2.3 Search Service for Heterogeneous e-Learning Resources

This work [9] carried out as a part of the project within the UK Joint Information Systems Committee (JISC) aims to provide a search service for the E-Learning Framework (ELF - <http://www.elframework.org>). ELF is a collaborative initiative by various research institutes to define a Service Oriented Architecture (SOA) for developing e-Learning services.

Huge volumes of e-Learning resources are produced across various institutions, but an efficient way of sharing and distributing is still under research. ELF was introduced to provide a framework for developing loosely coupled web services which can deliver certain specific services like library management, student assessment, e-Learning search engines etc. These services could be used in combination by techniques like “service composition” to build an overall functional e-Learning application.

The work identifies the importance of using Service Oriented Architecture for providing e-Learning services and has proposed a service for searching heterogeneous e-Learning resources. The objective is to provide a simplified and unified search interface to search various e-Learning repositories. The user can enter the search query and receive hits from multiple sources which otherwise would require to use the individual search interface to discover the resources.

The d+ search service discussed in this work performs “*federated search*” as defined in the common services of ELF. Federated search deals with providing a single search interface which is interoperable across different repositories. The following are some of the repositories that could be searched using the search service implementation, (taken as in from the paper)

- ✓ “Bibliographic databases like Z39.50, SRW/U.
- ✓ Open-source institutional repositories: Dspace
- ✓ Learning object repositories: IntraLibrary
- ✓ Commercial search engines and catalogue services like PudMed, Google, Amazon and O’Reilly Safari e-reference library”

The search service which is a web service is deployed in the Tomcat web container and communication takes places through SOAP / REST protocols and information exchanged using XML files.

Comparison with NEEDLE

This work closely relates to the current thesis on NEEDLE. Both these applications aim at providing an open architecture for interaction with the external e-Learning repositories. More specifically, both of them use a Service Oriented Architecture to provide a simplified and unified search service (web service) capable of performing resource discovery on multiple heterogeneous repositories.

In this thesis work, we introduce the concept of “*Search Library*” which are implemented as web services and used to interact with external search engines like Google, Yahoo etc. Thus, the single search interface of NEEDLE is used to search its own repository and external repositories which the users can configure and select while searching.

With respect to the E-Learning Framework, NEEDLE could be considered as a valuable repository of e-Learning materials and through exposing more services, NEEDLE could be extended to interact with other applications based on ELF.

3.2.4 AdELE - Adaptive E-Learning through Eye Tracking

AdELE [7] is an interesting work carried out with the motivation for adaptive and personalized e-Learning. The researchers consider that user (learner) behavior is a very important factor to be considered while designing an e-Learning application. There is no point in designing an application which does not adapt according to the user needs and expectations. Personalization is required to tune the system according to the user’s level of learning and helps to set preferences.

The user interface for an e-Learning application plays an important role in grasping the attention of the learner and to make the learning process a comfortable one. Hence, the work discusses the importance of semantic adaptivity and personalization of user interfaces.

AdELE is an innovative framework in which the adaptivity is achieved through an eye tracking mechanism. The techniques combine real-time content tracking and real-time eye tracking on the user’s side and accordingly adapt the delivery of contents. The eye tracking system monitors the movement of user’s eyes on objects and areas of focus, time spent on each spot, frequency of visits, and sequences in which content is read etc.

The main component of AdELE framework is the Adaptive Semantic Knowledge Transfer Module (ASKTM) which acts as a bridge between the learner’s side and the course creator’s side. It collects real-time information about user’s eye

movements and related metadata with the help of User Information Module (UIM). The information exchange is achieved using XML standard to provide media and platform independence.

From the learner's point of view, the AdELE framework helps to adapt the contents and offers assistance based on learning or reading speed, progress level etc. For e.g. if a learner reads a sentence or a paragraph repeatedly, the eye tracking system detects this behavior. This would mean that the learner has difficulty in understanding the particular portion of content and hence some additional materials could be popped in or shown to understand better.

From the course creator's point of view, it gives an implicit user feedback and helps to include smart features of course generation and delivery.

Comparison with NEEDLE

AdELE framework brings to attention the importance of designing an adaptive and personalized e-Learning application. NEEDLE has scope for implementing personalized searching using user profiling techniques. Also, the search results could be ranked and presented according to level and preferences set by the learner. To achieve adaptivity and personalized search in NEEDLE, it is not required to use any special hardware devices, unlike AdELE where eye-tracking devices are used.

Chapter 4:

NEEDLE Re-engineering using

J2EE Design Patterns &

Hibernate

4 NEEDLE Re-engineering using J2EE Design Patterns & Hibernate

4.1 NEEDLE: Prototype

NEEDLE provides a web-based prototype for presenting a search screen for the user to enter a search query. The results are displayed as a list of hits with a brief summary of the text content, the link to particular part of the video lecture and audio track, the link to LODE system and the image of the presentation slide.

The data flow and control flow are straightforward. The query with other related meta-data are sent from the front screen to the business tier. The business tier then contacts the backend database to fetch additional data (like series name, event name, directory path etc.) required for performing the search. Then the actual search is performed on the Lucene indexes (created using the upload process). Finally, the business tier receives the search results, packs it and sends to the presentation tier for displaying.

From the above explanation it is clear that we use a 3 – tier or N –tier architecture for implementing the NEEDLE prototype. The existing prototype has been designed based on J2EE specification and implemented using J2EE components like Java Server Pages (JSP), Servlets etc. For database access, NEEDLE uses Java Database Connectivity (JDBC).

4.1.1 Need for Re-engineering

Though the existing prototype has all the essential features to support simple search functionalities, it has to be re-engineered to incorporate additional functionalities and for the following reasons,

1. To use better J2EE design pattern for proper separation of presentation-tier from the business-tier. We discuss more on design patterns in the next section.
2. Substitute JDBC with Hibernate technology for Object- Relational Mapping. More of Hibernate advantages are discussed in the coming sections.
3. To implement Advanced search functionalities to help the users to better refine the search query.
4. To introduce the concept of Search libraries based on Service Oriented Architecture, using which the NEEDLE system could be extended to search any online resource for e-learning materials.

In order to re-design and re-engineer the prototype, we investigate some of the basic concepts of J2EE, design patterns and Hibernate technology in this chapter.

4.2 Java 2 Enterprise Edition – J2EE

4.2.1 Evolution

Enterprise solutions have recently shifted the focus towards providing applications based on n-tier architecture that are secure, scalable, maintainable and available. Software giants like Sun Microsystems, Microsoft came up with their own specifications and products to provide the next generation enterprise applications. Sun's Java 2 Enterprise Edition (J2EE), and Microsoft's .NET Framework are successful in helping the developers to build applications that are web-based and frequently used to deliver e-commerce solutions.

The primitive distributed computing architectures were based on monolithic applications [10] containing all the functionality of the application in a single, large, and usually un-maintainable piece of code. This was the trend in the days of mainframes. All user input, validations, business logic, and data access were placed together and the entire application ran on a single system. These could not be exactly called as distributed applications. These systems were sufficient for running stand-alone applications that did not require any network connectivity. But, maintaining the code was very difficult. Any change in a part of a code might introduce bugs in other parts. Hence, these systems are not useful for modern scenarios.

Then the client-server systems based on 2-tier architecture were introduced with an intention to share data between multiple applications running in parallel on different machines. Typically, the client application initiates a communication with server by sending a request. The server processes the request and it is usually the database server which fetches data for the client. Usually these were designed as fat client systems [10], in which both the presentation logic and business logic reside on the client. The server just takes care of the database access. Alternatively in thin client systems [10], the client just has the presentation logic and the server takes care of the business logic and database access. This architecture can deliver a certain amount of scalability and flexibility within the system. But, still the scalability and maintainability is not good enough to satisfy the increasing business needs.

There was a development in an area called Component technology. A *component* [10] is a unit of functionality that can be used within a particular framework. Component frameworks greatly help to simplify the process of application development. When using a component framework, a container provides the components with certain standard services, such as communication and

persistence. These components interact with each other to participate in N-tier architecture.

In the latest N-tier architecture, each component is responsible for performing a particular function and interacts with other components to request for any needed functionality. Hence, in this system, there is a high chance for scalability and the any change required in a component can be done without much effect on other components. Therefore, we need N-tier, component based, web-friendly architecture to meet the business requirements of today's enterprise applications.

4.2.2 Introduction to J2EE

Java 2 Enterprise Edition - J2EE, is a development specification for producing secure, scalable, and highly-available enterprise applications [11]. There are servers that consist of various containers for deploying and running components. The J2EE compliant servers should follow the guidelines given by the specification. The containers will provide a defined set of services to the components. The J2EE specification provides a definition from which enterprise vendors can produce J2EE application servers on which J2EE-compliant applications can be deployed.

At this point, it should be noted that J2EE itself is not an application, but is a specification. The J2EE specification just defines components and services provided by them and the interaction between them. They do not tell anything about the logical architecture of runtime environments, address spaces or physical machines [10].

4.2.3 Components and Containers

The components participating in a J2EE environment are housed by J2EE compliant containers. The services that the container has to provide are given in the specification. Containers supply a runtime environment for the components. As such, Java 2 Platform, Standard Edition (J2SE) is available in each container. Application Programming Interfaces (APIs) in J2EE are also made available to provide communication between components, persistence, security etc. J2EE application server vendors are free to develop the containers following the specification. The following are the basic containers specified in J2EE [10]:

- Applet Container
- Web Container
- EJB Container

There are two types of components deployed and run on a J2EE application server:

- *Web components*—A Web component is responsible for the interaction with a Web-based client, such as a web browser. There are two types of Web components in J2EE—

- ✓ Servlet and
- ✓ Java Server Pages (JSP)

Both types handle the presentation of data to the user.

- *Enterprise Java Beans (EJB) components*—The EJB is responsible for implementing the business logic and data base access. There are three kinds of Enterprise JavaBeans components—Session beans, Entity beans, and Message-Driven Beans.

Figure 4.1 illustrates the J2EE logical architecture typically followed by enterprise applications.

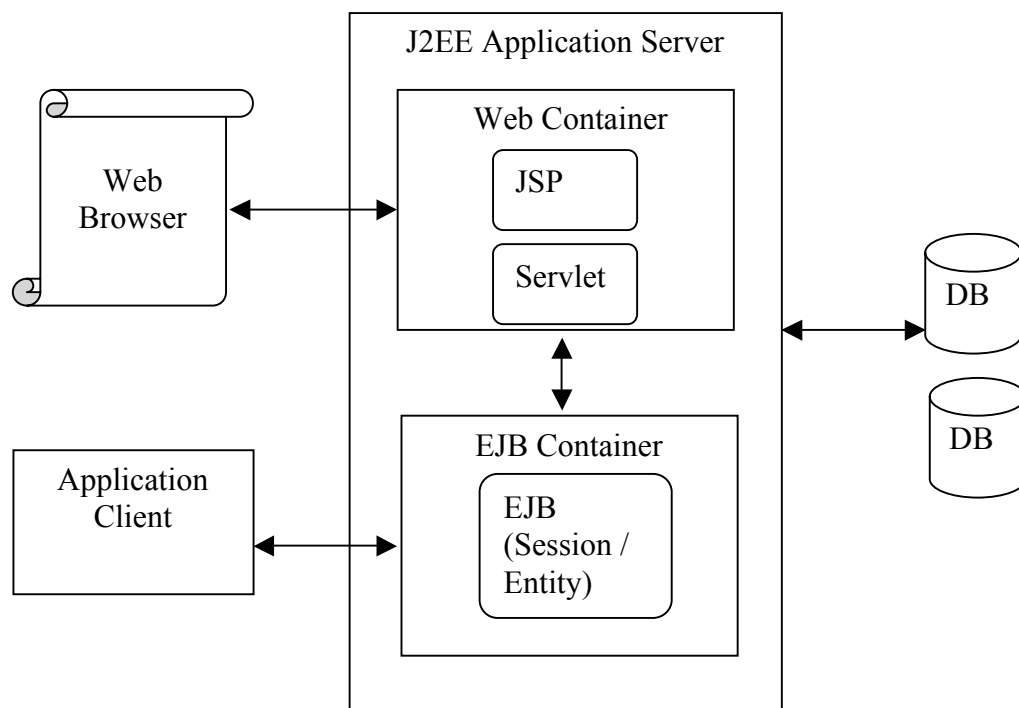


Fig. 4.1 J2EE logical architecture

4.2.4 Presentation Tier: Web Components

The web components part of J2EE is responsible for the presentation tier of the enterprise application. The presentation logic will decide which screens are displayed to the user in which order and the interaction with the business tier to carry out a particular business process. The client need not be always web based. The client could be a WAP enabled mobile or a standalone Java application with Swing interface. The way in which user input is received and information is displayed will depend on the type of client. Each type of client will require different presentation tier functionality to exchange and display information. The most common type of client for a J2EE application is a Web client that runs on a Web browser.

4.2.4.1 Java Server Pages (JSP)

Java Server Pages are responsible for dynamic creation of markup pages, like HTML, in response to a client's request [12]. It could be compared to famous Internet technologies like Active Server Pages (ASP) or ColdFusion.

A JSP consists of a two components called as tags (JSP tags) and scriptlets. The scriptlets contain the executable code to embed some dynamic behavior and also static markup like HTML. The server identifies the executable code in the JSP and executes it. The resulting page is delivered to the client. The advantage of using the embedded code is that, the extra static markup could be created in a dynamic manner and displayed to client.

The JSP tag is the basic component of a JSP page. It limits the sections of executable code. Scriptlets boosts up the JSP's processing power as it allows the logical code to be embedded inside sections of script [12]. Typically, the code is written using the Java programming language.

It is important to understand the role of JSP within a J2EE application and its interaction with other components. When a user requests for a JSP page for the first time, the JSP container converts the JSP into a Java file and compiles it. It is usually converted into a Java Servlet. The Servlet forwards the request to a business logic component, such as another Servlet, a JavaBean, or an Enterprise Java Beans. The component performs some action, such as accesses a database or processes the client's input, and returns a response to the Servlet. The Servlet passes this response to a JSP that generates the markup language that would be presented to the client.

4.2.4.2 Servlets

Servlets are responsible for receiving a HTTP request from a web-based client, process the request and give back an HTTP response to the client. Servlets use a request-response model and assists the web servers in processing the requests [10 & 11]. In the earlier systems, CGI scripts were used to provide such functionality. The following are the advantages of Servlets over CGI scripts:

- *Speed*—Servlets could be deployed as Java class files. The class file consists of Java byte codes, which are much faster than interpreted scripting languages, such as CGI - Perl.
- *Platform Independence*—Servlets can be executed on different servers and platforms as they are platform-independent classes.

One of the important advantages of Servlet is its ability to interact with other J2EE components. Just like JSP, the Servlets can communicate with another Servlet, JavaBeans, and EJB etc. Fig. 4.2 illustrates the interaction between client, Servlet and EJB container.

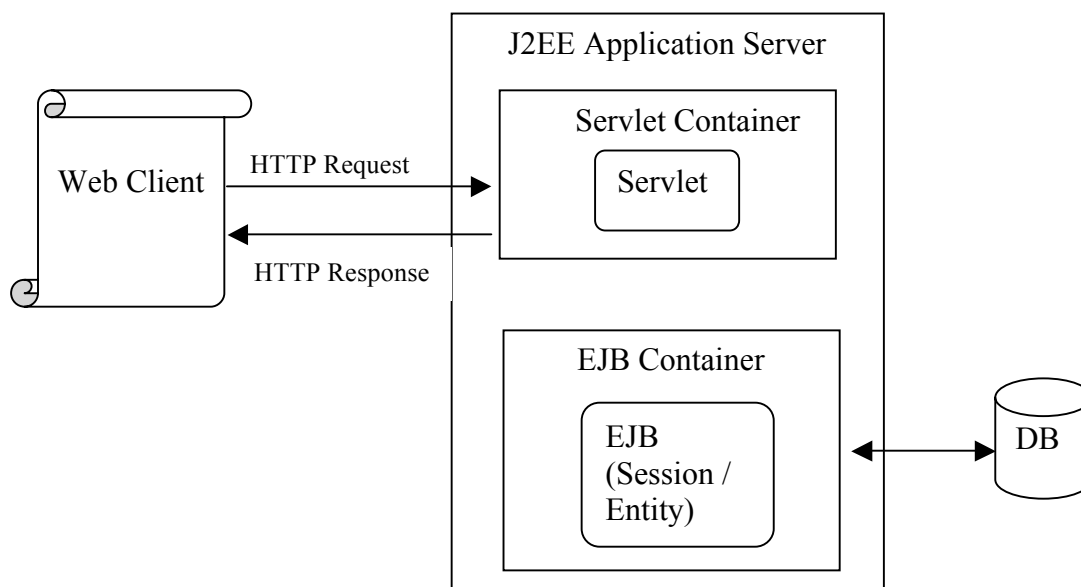


Fig 4.2 Interaction between client, Servlet and EJB

Servlet Life Cycle

The web container is responsible for the life cycle management of a Servlet [11]. The below steps are carried out by the container, when a request is directed to a Servlet [13].

- a) Loads the Servlet class.
- b) Creates an instance of the Servlet class.
- c) Init method is invoked to initialize the Servlet.
- d) Next, the service method like doPost() or doGet() is invoked by passing the request and response objects.

The Servlet destroy method is invoked to remove the Servlet and end the lifecycle.

4.2.5 Business Tier: Enterprise JavaBeans

In enterprise applications based on J2EE specification, the business tier is usually represented by Enterprise JavaBeans (EJB) [14]. Other components like simple Java objects could be used for the business tier, but EJBs are advantageous to encapsulate and share common business logic. Also the EJB container provides more services like transaction management etc.

There are few types of EJBs which are usually built together to carry out a transaction or a business process. They are

- ✓ Session beans – Stateful or Stateless,
- ✓ Entity Beans
- ✓ Message Beans

As we are not going to use EJB in NEEDLE, we just discuss the few basics of each type of EJB.

Session Beans

Session beans [14] are the simplest and the most commonly used EJB. A session bean usually consists of a set of common business functions. The Servlet or the client invokes the business methods on the Session bean. The business data or the transaction data need not be stored in the Session beans. The session bean may not hold any data or may hold data only for a particular session. If a session bean wants to access the data from a database, it can use JDBC calls or by using Hibernate. Alternatively, one can also use entity beans to access and manage persistent data. There are two types of Session beans,

- ✓ *Stateful Session beans* – This type of session bean remembers the conversation between the client and itself over a session. It maintains state. For each client, a separate, dedicated bean instance is created and used to maintain the state information.
- ✓ *Stateless Session beans* – This type of session bean does not maintain state. The bean instances for all the clients look alike and they do not store any information particular to a client.

Entity Beans

An *entity bean* is a persistent object that stores and manages data from a data base; its key responsibility is to fetch, maintain and update data from a database. A row or tuple in a table is represented by an instance of the entity bean and is identified by a primary key. The persistence of the entity bean is either managed by itself or by the container.

Persistence is handled in one of two ways:

- ✓ **Bean-managed persistence (BMP)** - The developer has to write code (as part of the entity bean) to manage persistence.
- ✓ **Container-managed persistence (CMP)** – The EJB container takes responsibility of managing persistence.

The advantage of using CMP Bean is that they are more portable across databases. It is also possible to maintain relationships among container managed persistence beans. Using this feature it is easier to perform queries that join multiple database tables.

In bean-managed persistence, if there is a change in the underlying database, the developer needs to modify the SQL source code of the entity bean to conform to the SQL implemented by the new database. Hence, maintainability of the code becomes difficult.

Client Interaction with EJB

Two mechanisms found in J2SE—namely Remote Method Invocation (RMI) and the Java Naming and Directory Interface (JNDI) – are used by EJBs to facilitate their interaction with the client. When an EJB is written, the functionality it offers to

clients is defined as an RMI remote interface. The location of the deployed EJB is registered in the naming directory service.

The JNDI is used by the client to look up the location of the EJB. The client then uses the EJB's home, to obtain an instance of the EJB. This could be compared to the usage of 'new' to create an instance of a local object. When the client has a reference to the EJB, it can invoke the business methods exposed by the EJB. These steps are shown in the Fig. 4.3

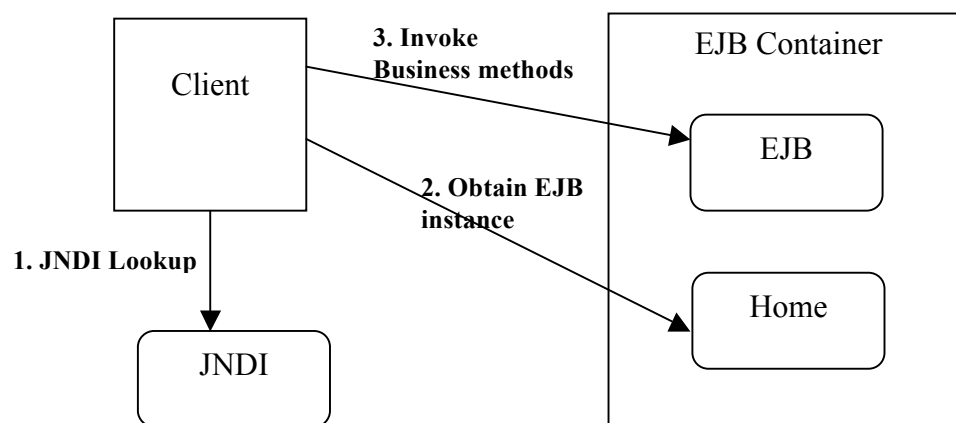


Fig. 4.3 Client interaction with EJB using RMI and JNDI

4.3 Design Patterns

A pattern is defined as a proved solution to a problem with respect to a context [15]. In simple terms, a pattern is a reusable idea on how to solve a particular problem found in the domain of architecture or design of software systems. The concept of patterns were initially used in construction of buildings, machineries etc. Now, it has been successful in software systems also. The important point to note about patterns is that they are proven solutions. They could be employed in scenarios without much research and investigation.

4.3.1 J2EE Patterns

As discussed in the previous sections, J2EE specification is based on components interacting together to deliver an enterprise solution. When many components are employed in an application, there is often confusion and misunderstanding in the usage. Hence, the concept of patterns would be beneficial in designing J2EE applications. The purpose of using these patterns is to know the correct or proper ways of using the J2EE technologies to address common problems found when creating modern, enterprise applications.

A 3-tier business model is the most common J2EE architecture. Though the design pattern could be extended for N-tier applications, as of now the current J2EE patterns are mainly targeted at a 3-tier, Web-based business system [15]. The 3 tiers are usually classified as the presentation (usually Web based), the middle or business tier, and the data access tier.

The most common J2EE architectures roughly follow the Model-View-Controller (MVC) pattern [15]. In terms of J2EE components, this translates into entity EJBs / Hibernate objects for providing a data model, Servlets and JSP providing the view, and Session EJBs providing the controller or business logic. Dividing responsibilities in this fashion delivers a lot of flexibility. An example of this is that the model and controller (the data and the business logic) can be combined with a variety of views to expose the same functionality to different clients like web-based, WAP clients etc.

4.3.2 Model View Controller (MVC) Pattern

By using the Model-View-Controller (MVC) pattern to a Java 2 Enterprise Edition (J2EE) application, it is possible to separate core business model functionality from the presentation. This sort of separation allows multiple views to share the same enterprise data model.

The MVC architecture was first used in Smalltalk [15]. It was used for receiving user input, processing and output for GUI model. However, it was found useful to apply these concepts into the area of multi-tier enterprise applications.

- **Model** - The business data is represented by the model. The data model could be used as a temporary (session based) storage of data between the various components of the enterprise application. It could also represent the persistent data fetched from a database and manages the creation, updation and deletion of such data. The common components used as data model are JavaBeans, EJB entity beans, Hibernate objects etc.
- **View** -The responsibility of 'view' is to present the contents of a data model i.e. the business data. It fetches the enterprise business data through the model and determines the presentation of the data. Whenever there is a change in the model, the view should maintain consistency in its presentation. For this purpose, the view can either use a push model or pull model. In a push model, the view registers with the model to receive notifications in case of any change. In pull model, the view needs to check with the model whenever it needs the latest data. Java Server Pages (JSP) and Servlets are usually used as view.
- **Controller** - The controller decides on which action to perform based on the user input or based on business logic. User requests in web based clients come in the form of HTTP GET or POST operations. Based on the user interactions and the result of the business logic, the controller selects an appropriate flow.

4.3.3 MVC Pattern in NEEDLE Prototype

The NEEDLE application presents the e-learning materials through a web-based user interface. The existing prototype is based on J2EE architecture using web components like JSP, Servlets and used Java Database Connectivity (JDBC) for accessing persistent data.

The MVC pattern is used in the implementation of the prototype. The presentation-tier in NEEDLE consists of a search screen where a user enters the query to search in the e-learning materials. The results are displayed as a list of hits with a brief summary of the text content, the link to particular part of the video lecture and audio track, the link to LODE system and the image of the presentation slide.

The business – tier i.e. the actual search process is implemented using a Plain Old Java Object (POJO). This in combination with other utility classes perform the search, receives the hits results, packs the results in JavaBeans object and delivers it to the Servlet. The Servlet controls and selects the particular screen or view (JSP) to be displayed.

The data access was previously implemented using JDBC. For advantages to be discussed in the next section, we have chosen to implement Hibernate technology instead of JDBC for data access.

To summarize and map the model, view and controller for the NEEDLE system,

- ✓ **Model** – NEEDLE system uses JavaBeans to store and transfer business data between JSP, Servlet and the business tier. For storing persistent data, Hibernate objects are used.
- ✓ **View** – Java Server Pages (JSP) are used to present the search results and the data represented by data model. Servlets help to prepare the data to be presented in the JSP.
- ✓ **Controller** – Servlets and the Java objects are used for business logic and they act as controllers for the selection of presentation screen, data flow and control flow.

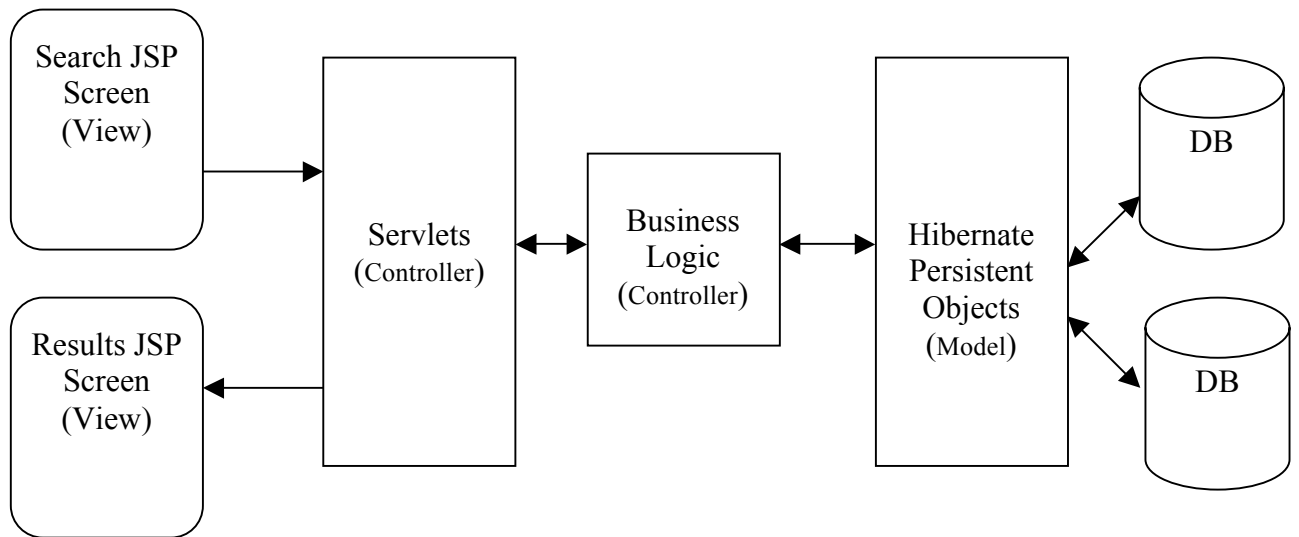


Fig. 4.4 NEEDLE MVC Pattern

4.4 Data Access Layer

The data access layer in NEEDLE was implemented using Java Database Connectivity (JDBC). It is easy to achieve database access using JDBC and efficient for small applications.

4.4.1 Java Database Connectivity (JDBC)

JDBC allows developers to use SQL (Structured Query Language) to open a database connection, perform query operations and to update the database objects like tables etc [16]. JDBC API uses database specific JDBC drivers. It assists Java developers to interact with different RDBMS and access table data through Java application without learning much RDBMS details. The following are the steps in interaction between Java application and RDBMS using JDBC:

- 1) **Load Driver** - RDBMS specific JDBC driver is loaded which actually communicates with the database.
- 2) **Open Connection** – Database connection is established which is then used to send SQL statements and fetch results.
- 3) **Create Statement Object** – JDBC Statement object which contains the SQL statement (query) is created.
- 4) **Execute Statement** – The Statement object is executed i.e. the SQL query is executed which returns ResultSet object. ResultSet contains the records or tuples of database table as a result of SQL query.
- 5) **Process ResultSet** – The ResultSet is iterated and the individual records are fetched and processed.
- 6) **Close the connection** – Finally the database connection is closed.

JDBC could use either of the two following architectures to communicate with database:

- 1) The database specific driver connects to database and executes SQL statements. The driver receives the result set and sends it to driver manager which finally forwards it to the application.
- 2) Alternatively, the JDBC driver interacts with ODBC driver. ODBC driver executes SQL query and sends back the result to JDBC driver.

4.4.2 Hibernate

Hibernate provides an Object-Relational Mapping (ORM) solution [16] for JAVA developers to use object-oriented idiom – including association, inheritance and polymorphism to develop persistent classes. In simple terms, Hibernate provides a solution to map database tables to a class and the columns to a data member of the class. It copies the database data to a class. In terms of the other way, it saves objects to the database.

Most of the OO programmers are quite comfortable and tend to work best when working with Java Objects, and they take time to write good SQL code. Hence, there is a mismatch in understanding the mapping between the objects and the corresponding database tables. Hibernate technology helps to bridge this gap.

Hibernate Architecture

Hibernate:

1. Opens connection to database. It uses the Hibernate properties configuration file (*hibernate.cfg.xml*)
2. Takes care of the conversion of HQL (Hibernate Query Language) statements into SQL statements.
3. Receives the result set.
4. Performs mapping of the database specific data to Java objects by using the *hbm.xml* file which defines the mapping between database field properties and the Java object.

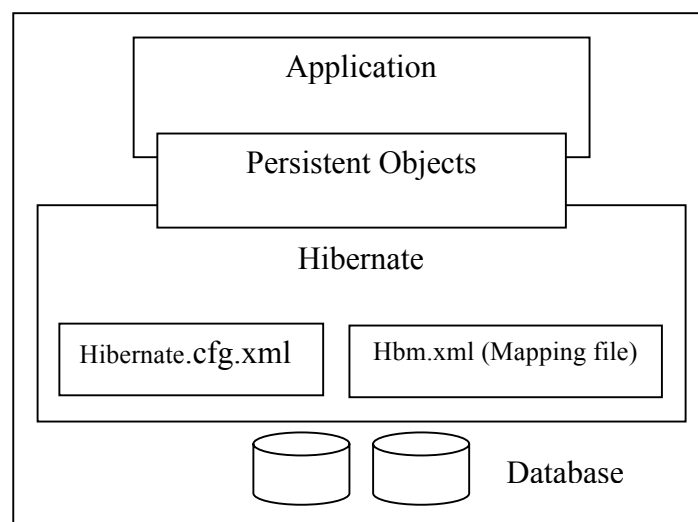


Fig. 4.5 Hibernate Architecture

4.4.2.1 Hibernate Vs JDBC

1. Object - Relational Persistence

It is complicated to work with both Object-Oriented language and Relational Database in JDBC because there is mismatch between data representation in objects and RDBMS. So with JDBC, it is the responsibility of the developer to write code to map between the two representations. With Hibernate, the developer just needs to define the mapping between the database fields and Java object members in the XML file.

2. Database Independent code

JDBC relies on native SQL. Developer needs to know the syntax for writing the SQL query for the specific database. Hibernate provides a powerful query language called Hibernate Query Language (HQL) that is independent of any database and uses SQL like syntax. Using HQL it is also possible to write complex queries.

3. Maintainability

With JDBC, if there is a change in the table or database then the developer has to change the whole code written in Java to map table-to-object/object-to-table. But with Hibernate, this mapping is present in the XML file. It's just required to change only the XML file if there is change in Database.

4.4.2.2 Hibernate Vs Entity EJB

1. Simple Java Code

Hibernate persistence objects are implemented using Plain Old Java Objects (POJO) unlike EJBs which require special naming conventions, follow specific architecture, and inherit classes and methods to supports RMI etc.

2. Application Server not required

EJBs are deployed in an EJB container which is provided by Application Server. Such architecture is required only for heavy-weight applications supporting huge volume of client requests. For simple applications, Hibernate proves to be a better choice.

4.4.3 Hibernate Implementation in NEEDLE

The data access layer of the NEEDLE system has been re-engineered with Hibernate technology to provide Object Relational Mapping and the advantages discussed in the previous sections.

The *hibernate.cfg.xml* file contains the database specific details like driver class, database URL, user name, password, schema name, mapping file name etc. NEEDLE uses MySQL as backend database. The Hibernate engine uses this configuration file to establish connection with the backend.

```
<hibernate-configuration>
<session-factory>
  <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
  <property name="hibernate.connection.url">jdbc:mysql://localhost/looodle</property>
  <property name="hibernate.connection.username">root</property>
  <property name="hibernate.connection.password">root</property>
  <property name="hibernate.connection.pool_size">10</property>
  <property name="show_sql">true</property>
  <property name="dialect">org.hibernate.dialect.MySQLDialect</property>
  <property name="hibernate.hbm2ddl.auto">update</property>
  <!-- Mapping files -->
  <mapping resource="looodle.hbm.xml"/>
</session-factory>
</hibernate-configuration>
```

Fig. 4.6 Hibernate Configuration XML file

The *looodle.hbm.xml* file contains the mapping between the Hibernate Java objects and the corresponding database table fields. The mapping contains details about primary key, data type etc.

NEEDLE contains 11 tables namely, 'actor', 'document', 'document_event', 'document_series', 'event', 'event_actor', 'event_type', 'event_view', 'serie', 'serie_type', 'view'. Hibernate Java class is created for each of the table with the columns as data members and corresponding getter and setter methods.

The *LoodleDBIntf* interface is implemented by *LoodleHibernateImpl*. The methods in this class use Hibernate Query Language (HQL) to access the database and the results are iterated and processed using the Hibernate persistent objects. Hence, from the business logic class any call to database is done through *LoodleHibernateImpl*.

LoodleJDBCImpl class also implements *LoodleDBIntf* and contains the old JDBC code for accessing the database. This class is no more used but is just maintained for future reference.

Chapter 5:

NEEDLE Automation Tool for Uploading E-learning Materials

5 NEEDLE Automation Tool

5.1 Objective

As discussed in the earlier sections, the NEEDLE system aims at presenting multimodal e-learning materials through a web based interface. Importance has been given to the design of efficient search functionalities including simple search and advanced search features.

The input to the NEEDLE system is provided by the LODE system which integrates the presentation of the video lectures, presentation slides and assigns timestamp tags for easy navigation through the presentation. These video / audio lectures, slides, documents are the backend data for NEEDLE. When a user searches for any e-learning material, the query is searched in the audio transcript of the video lectures or the text content of the slides / documents.

In order to perform these searches, the data from the LODE system has to be pre-processed and placed in specific directory structures. The pre-processing steps include splitting of videos into smaller parts, extraction of audio tracks from video, extraction of key frames (images) from the video at regular intervals. The audio transcripts are also pre-processed to index the text data using Lucene API and the indexes are also placed in specific directory structures. The directory structure information for the video lectures and slides are stored in database for each course / lecture.

All the above mentioned activities were previously done manually. The input data from LODE system in archive format (zip format) were extracted and manually placed in the directory structures. The videos were split and audio extracted using third party tools. Then the database updates were performed individually. All these were time consuming and error-prone activities.

Thus, an automation tool to upload the videos was necessary to perform the above mentioned activities with least amount of user interaction. The user could use the upload tool to provide archived data content as input and the tool does all the activities in a sequential manner. This results in a user-friendly interface for uploading the e-learning materials to make them searchable in the NEEDLE system. More details about the requirements, design and implementation are discussed in the following sections.

5.2 Scope

In this section, we will discuss about the usage scope of the NEEDLE upload tool. As discussed in the earlier section, the tool is an essential component for the NEEDLE system to automate the process of uploading the e-learning materials, process them and make them searchable. Therefore, this tool is designed as a stand alone application that could be executed whenever a new set of video lectures or slides are available for the NEEDLE system.

The upload tool is designed as a Java Swing application. The tool is packaged with all the necessary libraries for its execution. On running the tool, the user is prompted to input the archived data content from the LODE system, the audio transcript file of the video lecture and the corresponding presentation slide. After the execution of a sequence of steps, if everything proceeds successfully, then the new materials uploaded would be available for searching from the NEEDLE web interface.

5.3 Requirements

The first step to design a tool would be to analyze and gather the specific detailed requirements. The following are the functional requirements.

1. The input to the tool would be the following:
 - i. Data Archive – Archive (.zip format) containing the video and audio lectures in the specific directory structure (as discussed in the implementation section).
 - ii. Audio Transcript – The transcript of the video lecture in .trs format.
 - iii. Presentation slides – The PowerPoint presentation slides (.ppt format) used in the video lecture.
2. The series details (i.e. the course details) and the event details (i.e. the lecture details) should be inserted in the database.
3. The archived data file should be unpacked (zip contents should be extracted).
4. The unpacked data should be placed in the specific directory/folder structure consistent with the database entries.

5. The audio transcript file has to be parsed and the text data should be pre-processed for indexing.
6. The processed text data of the transcript should be indexed using Lucene API.
7. Similarly, the presentation slide has to be processed and the text data should be indexed using Lucene.
8. The unpacked video data should be split into smaller parts at regular intervals (every 10 minutes).
9. The audio tracks should be extracted from the video lectures and stored in the specific directory structure.
10. The key frames (images) are extracted at regular intervals from the video lectures.

5.4 Design

The upload tool is designed as a stand alone Swing application. The important design consideration is the sequence of activities that are carried over by the tool. The objective is to prepare the input video lectures and make it searchable from the NEEDLE web interface.

In order to understand the sequence of inter-related activities, we have used UML Sequence diagram. Sequence diagrams [21] are used during requirements analysis and design phase for the following,

- Use case descriptions could be refined.
- More classes and objects could be identified.
- Represent the sequence of activities on time line.

Next we discuss about some of the key components of sequence diagrams. This would help us to understand their usage in the design of the tool.

- **Actors** - represent external entities initiating the sequence

- **Objects** - specific entities involved in the collaboration (rectangles) and exchange of messages.
- **Messages** - exchanged during interactions (labeled solid arrows)
- **Lifelines** - life time of an object (dashed lines)
- **Activations** - time during which an object is performing an action (vertical bars)

5.4.1 Sequence Diagram for Needle Upload tool

In our case, the external actor interacting with the system is the user. The user provides the input to the *NeedleUploadTool* object. The *NeedleUploadTool* object accesses the database through the *LoodleHibernateImpl*. The *CreateIndex* object is used by the *NeedleUploadTool* to create the Lucene Index. The *MP4Box* object is used to split the videos and to extract the audio from the video lectures. The *QuickTime* object is used for extracting the key frames.

The complete interaction between the actors and the objects are shown the following sequence diagram.

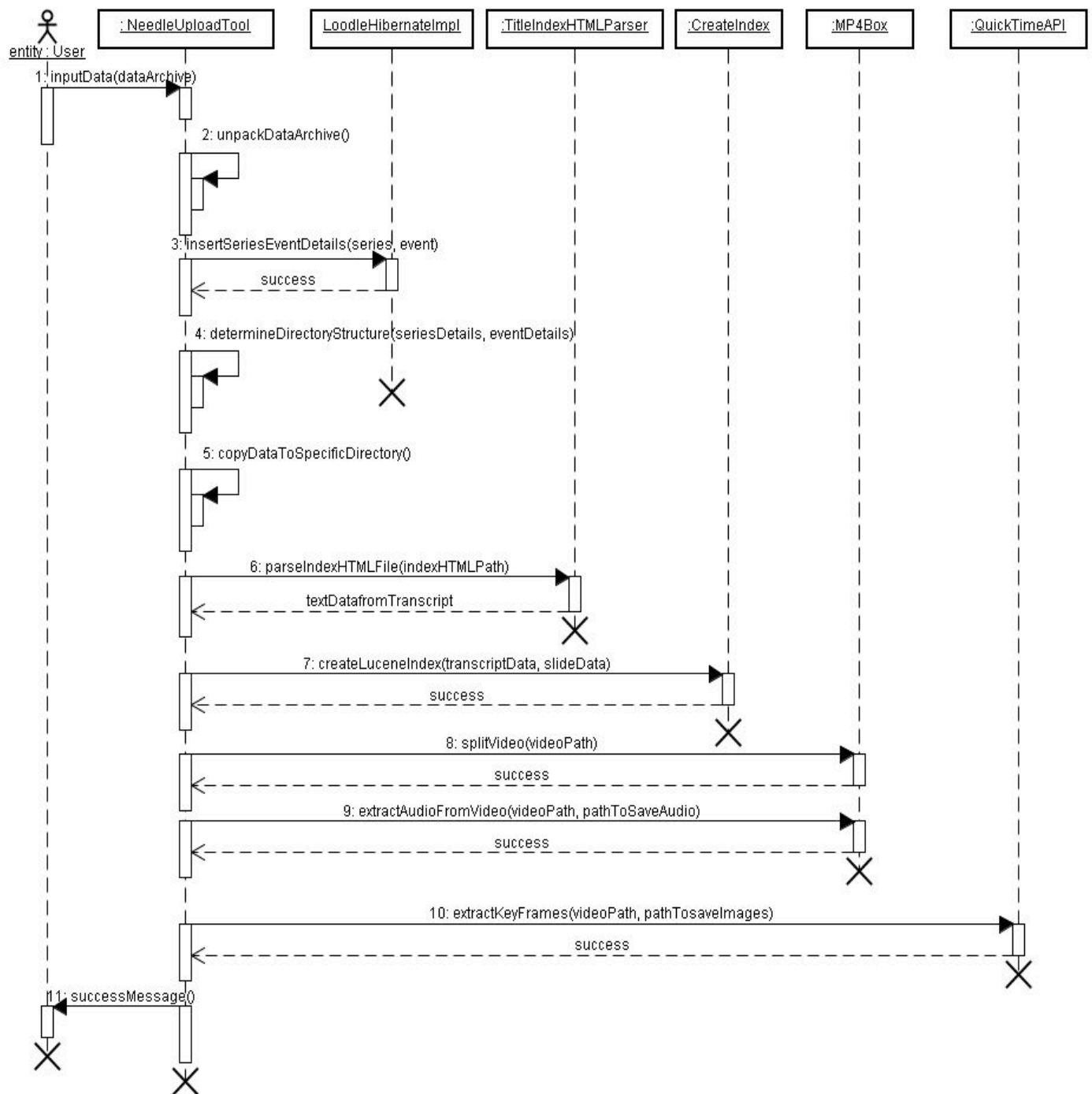


Fig. 5.1 Sequence diagram showing the sequence of activities by the *NeedleUploadTool*

5.5 *Implementation*

Now that we have gathered the requirements and designed the tool, the next step would be to make a study of the available technologies to implement the tool. In this section, we discuss about Lucene, QuickTime API and MP4Box which are the important technologies used for the development of the automation tool.

5.5.1 Lucene API

5.5.1.1 Introduction

The Lucene is a Jakarta open source search API that is used to create and search text data [17]. It's widely used in text-based search applications. Any text information can be stored – more specifically indexed, using Lucene and then find it later based on various search criteria. Recently, many enhancements have been added (contributed) to Lucene and now it can be used to index Word documents, PDF files, XML, or HTML pages. Thus, it is very flexible to use Lucene for web based search applications. Lucene uses fuzzy logic to search the indexed text [18]. It provides a basic framework that the developers can use to build full-featured search into the web sites. It should be noted that Lucene is specifically an API and not an application. This means that all the complex parts have been done, but the easy programming has been left to the application developers. The developers have to decide on the various options and customizations provided by Lucene to build a search application of their choice.

5.5.1.2 API Overview

Before we discuss about the Lucene indexing and searching, it's important to understand the important classes in the API and their purpose. The following table [18] summarizes the classes that are frequently used,

Class	Description
<i>Document</i>	The <i>Document</i> class represents a document in Lucene. Both index objects and search result objects are <i>Document</i> objects.
<i>Field</i>	The <i>Field</i> class represents a portion of a <i>Document</i> . The <i>Field</i> object consists of section name and data.
<i>IndexWriter</i>	This class is used to create Lucene index and maintain it.
<i>IndexSearcher</i>	This class is used for searching the Lucene index.
<i>QueryParser</i>	This class takes in a query string, parses it and generates a <i>Query</i> object.
<i>Analyzer</i>	Before forming an index, the <i>Document</i> object should be processed and the text has to split into smaller tokens. This task is done by the <i>Analyzer</i> . The developers could provide many implementations of this <i>Analyzer</i> class. The <i>StandardAnalyzer</i> is the commonly used implementation.
<i>Hits</i>	The <i>Hits</i> class contains the results of the search. Each result is a <i>Document</i> object.

5.5.1.3 Indexing

The major component of Lucene is an *Index* [18]. Any text information could be fed into the Index, then perform searches on the Index to get results out. The Index could be built using an *IndexWriter* object. An *IndexSearcher* object is used to perform searches on the Lucene index.

Index consists of *Document* objects. A Document object is a collection of *Field* objects (name/value pairs). The data is processed and is converted into Document objects and are inserted into the Index. The text data are read from files or database etc., broken down into chunks and stored in the Document as *Field* objects. Finally, the Document is written into the index using the *IndexWriter*.

A *Field* object consists of a name and a value, and in addition, it also contains three Boolean values. These values are used for deciding whether or not the value will be indexed for searches, tokenized prior to indexing, and stored in the index.

IndexWriter

IndexWriter [18] takes a *Document* object as input, feed it through the Analyzer, and creates an index. The *IndexWriter* is instantiated with parameters for location of the index files and the Analyzer that we want for processing the tokens. Then the Documents are fed into *IndexWriter.addDocument ()*. Finally, the created index consists of a set of data files that the *IndexWriter* creates in a location defined by a path string.

5.5.1.4 Analyzers and Tokenizers

The Analyzer takes a string of text and cleans up and outputs a stream of tokens. The tokens are usually words from the text content of the string, and it gets stored in the index.

Each analyzer includes one or more tokenizers and may include filters. The tokenizers are used for defining the actual rules for breaking up the text into words (e.g. white spaces, commas etc). The filters do any further processing the tokens, usually eliminating any stop words like "the", "an", "a", etc.

Lucene provides an Analyzer abstract class, and three implementations of Analyzer. The following table summarizes them:

SimpleAnalyzer	<i>SimpleAnalyzer</i> uses a Tokenizer that simply converts the input into lower case.
StopAnalyzer	<i>StopAnalyzer</i> converts the input into lower-case and also ignores the common stop words like 'a', 'an', 'the' etc.

	StopAnalyzer comes with a set of stop words, but developers can instantiate it with the own array of stop words.
StandardAnalyzer	<i>StandardAnalyzer</i> in addition to lower-case and stop-word filters, it includes features like removal of apostrophes (‘) and periods from abbreviations (i.e. "H.B.A" becomes "HBA").

There are many language specific analyzers developed by Lucene users. For e.g. Italian or German analyzers have support for eliminating stop words in the respective languages. It would be a straightforward approach to implement an analyzer for our specific requirements if we can't find a suitable analyzer.

5.5.1.5 Deleting Documents from Index

One way of deleting a Document is using the document number in the index. The Document number could be obtained by iterating over the Hits and this number is used in the IndexReader.deleteDocument (docNum) method. That will delete the document numbered 'docNum' from the index. Once a document is deleted any attempts to read its field with the document method will result in an exception.

We could also delete (one or more) documents that contain a specific term. This is done by using the IndexReader.deleteDocument (term) method. Since this method could delete a variable number of documents, this method returns the number of documents deleted.

5.5.2 QuickTime API

QuickTime API [19] provides rich set of functionalities to develop multimedia applications without any knowledge of a particular media format and specifications. For e.g. it could be used to develop an application that gets as input a set of JPEG images and outputs a slideshow movie, for splitting video files etc.

5.5.2.1 Multilevel API

The QuickTime API comes with more than 2000 functions, classified into tool sets for particular tasks, with API and data types to support various tasks.

The QuickTime API could be for the following purposes:

- It could be used to simply open and play movies, letting QuickTime handle all the file and format conversion, data buffering, loading and unloading of component, memory management, and even the user interface.
- It could be used for editing movie or track segments. For e.g. modifying the play rate, frame rate, compression standard, rearranging playback etc
- Movies could be generated programmatically from a set of input images.
- Developers could also create new QuickTime components to support features such as compression algorithms, new media types, media recording devices, output devices, or data sources.

Developers use the API's through their favorite programming languages to build applications with advanced functionalities. It is possible to invoke the QuickTime API's functionalities using C, C++, Objective C, or Java. There is also support for JavaScript, Visual Basic, C#, and .Net frameworks.

In most cases, developers invoke the QuickTime API from procedural C programs written in C or C++. There are some key differences in the implementation of the QuickTime API for Windows and for Mac OS. There are some Mac OS- only specific functions. The documentation of any function in the *QuickTime API Reference* indicates whether that particular function is included in Windows version or not.

The QuickTime for Java (QT Java) is the Java API for QuickTime. It is possible to make calls to the procedural C parts of the QuickTime API from Java.

5.5.2.2 QuickTime for Java

QuickTime for Java [19] is the Java API to access QuickTime, which helps the developers to create Java multimedia applications that take advantage of the power of QuickTime on both Macintosh and Windows. It has proved to be very helpful for Java developers to take advantage of the rich media capabilities QuickTime provides, including the support to play QuickTime movies, edit and create movies, and work with sophisticated 2D and 3D graphics.

QuickTime for Java consists of two layers:

- The core layer helps to access the complete QuickTime Application Programming Interface (API)
- The application framework layer is used for easy integration of Java applications with QuickTime capabilities.

The QuickTime for Java API is implemented as a set of packages grouped based on the functionality, usage and also the support for specific media types.

OO developers find it helpful to use QuickTime for Java API as it provides an object oriented approach for the QuickTime API and a binding of QuickTime's native calls into Java method calls using Java Native Interface (JNI). It supports garbage collection and does not use any pointers or other features that are common in a C-based API.

5.5.2.3 QuickTime API usage in NEEDLE

The automation tool for the NEEDLE system has to process the video lectures given as input and perform some media specific operations. QuickTime API is found helpful for these requirements. For e.g. QuickTime for Java could be used for extracting key frames (images) from the video lectures at regular intervals. The video file is loaded as *Movie* object and the image at a particular time is extracted using *Movie.getPict (int time)* method. This method returns a *Pict* object containing the image in .pict format. In order to convert from .pict format to .jpeg format, we make use of the *GraphicsExporter.doExport ()* method by passing the parameters as the format to be converted (in our case it would be “JPEG”). It could also be used to extract the audio content from video and for splitting the video into smaller parts.

5.5.3 MP4BOX

MP4BOX [20] is a simple command line utility to work with Mpeg4 (.mp4) media files. It could be used for generating new mp4 movie, editing, extracting audio, splitting the video into smaller portion and many such functionalities.

The tool was found to be very simple and user friendly. It is also easy to invoke this command line utility from a Java program. Hence, it has been used for two functionalities required by the upload tool, namely: splitting of video and extraction of audio tracks.

Splitting of Video lectures

As the running time of a video lecture is quite long, it is important to split it into smaller parts to display as search results. This would help the user to view the exact position of the occurrence of the search term in the video lecture. Also, the performance would be much better in loading a smaller video on the web interface. Hence, splitting of videos is an important functionality.

The following command is used to perform the operation through MP4BOX

```
C:\> <Name of the MP4Box .exe file> -split <split time interval in seconds> <video file path>
```

E.g. C:\> MP4BOX.exe -split 60 lesson1.mp4

Audio Extraction

It is also required to extract the audio tracks from the video lectures to display as search results. Users who just want to listen to the audio can do so without using the video lectures.

The following command is used to perform the operation through MP4BOX. The number of the track to be extracted is given as input. In mp4 format audio track is number 5.

```
C:\> <Name of the MP4Box .exe file> -single <track number> <video file path>
```

E.g. C:\> MP4BOX.exe -single 5 lesson1.mp4

content

data

content_frame

collection of images

5.5.4 NEEDLE Implementation

title.html

Before going into the details of the class implementation, we first discuss about the directory structure that is expected for the input archive file.

movie.mp4

Input Data Archive

index.html

common

The user is supposed to provide as input the video lectures and other meta-data that is produced by the LODE system. These files should be packaged as a .zip file and the directory structure present inside this archive (.zip file) is very important. It is assumed that the input file obeys this structure and the process is executed. Any difference in the structure from the one discussed here, would result in failure of the process. The names in bold and italic should be present as given.

XYZ.zip (Any name .zip)

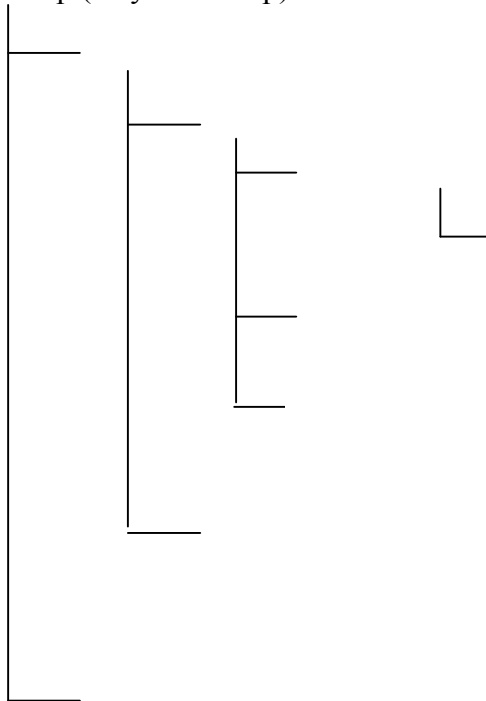


Fig. 5.2 Directory structure of input archive file

Classes and Methods

Now we discuss some of the important classes and methods used in the implementation of the tool.

NeedleUploadTool

The *NeedleUploadTool* is the main class which inherits the swing *JFrame* class. It displays a frame for receiving the user input. It includes all the key methods for processing of the e-learning data. The following table briefly summarizes the important methods:

Method Names	Description
processUpload()	This method is the root method which in turn calls the other methods. It first unpacks the archive, inserts the records into database, then calls the corresponding methods to create Lucene index, split the videos, extract audio and to extract key frames.
parseTitleIndexHTML()	This method uses the <i>TitleIndexHTMLParser</i> object to parse the <i>title.html</i> and <i>index.html</i> file to fetch the series name, event name and the date of the event. These details are needed for inserting into database and to determine the directory names to place the video, audio, slides and the indexes.
cutVideos()	This method uses the <i>MP4Box</i> tool to split the videos into smaller parts.
extractAudio()	This method uses the <i>MP4Box</i> tool to extract audio tracks from the video lectures.
captureKeyFrames ()	This method uses the QuickTime API to extract key frames from the video lecture at regular intervals of time.
createIndex ()	This method calls the <i>CreateIndex</i> object to create the Lucene index for the transcript file and for the presentation slides.
insertSeriesDetails ()	This method uses the <i>LoodleHibernateImpl</i> object to insert the series details for a new series
insertEvent ()	This method uses the <i>LoodleHibernateImpl</i> object to insert the event details for a new event
setPathStringForData ()	This method is used to determine the directory path for placing the data. The path string is created by using the series name, event name, event id, series id etc.

CreateIndex

This class contains methods for creating the indexes for the text data of the audio transcript and the presentation slides.

TitleIndexHTMLParser

This class contains methods for parsing the *title.html* to fetch the series name; event name and *index.html* file to fetch the date of the event.

LodeIndexParser

This class contains methods to parse the *index.html* to fetch the timestamp, slide number for inserting into the *event_view* table.

ExportPPT

This class interacts with the *PPT2XML* class and *XML2TXT* class to convert the input PowerPoint slide into an xml document and then into a text document. Finally the test data is indexed using Lucene.

The above classes and methods are some of the important ones. The detailed information about the classes and complete method signature, return types and API documentation is found in the Javadoc for the upload tool.

5.5.4.1 Screen Shot

NEEDLE Upload Tool

File Edit Help

Data File(.zip) ktop\Lezione_9.zip **Brow...** Edition Second

TRS File (.trs) i\Desktop\lez09.trs **Brow...** Series Type Stand-alone Event ▼

Presentation... ktop\lezione_9.ppt **Brow...** Event Type Lecture ▼

Language Italian ▼

Submit

Screen.1 NEEDLE Upload Tool

- ✓ **Data File (.zip)** – The archive files containing the video lectures and related meta data taken from the LODE system.
- ✓ **TRS File (.trs)** – The audio transcript of the video lecture.
- ✓ **Presentation Slide (.ppt)** – The PowerPoint presentation slides for the corresponding video lecture.
- ✓ **Edition** – The edition of the lecture series or course.

- ✓ **Series Type** – User can select either “Stand-alone event”, “Summer school”, “Seminar series” or “University course”.
- ✓ **Event type** – Either “Lecture” or “Seminar”.
- ✓ **Language** – Either “Italian” or “English”.

Chapter 6:

NEEDLE Advanced Search Interface

6 NEEDLE Advanced Search Interface

6.1 Overview

The existing NEEDLE prototype provides a simple search interface for the users to quickly type in their query and get the results. Although this search interface is simple and efficient, it would be better to provide an advanced search interface with more options to help the user to refine the query and narrow down the search results. Most of the commercial search engines and search applications provide this option. Such additional options for searching are very important for searching heterogeneous multimedia data like, videos, slides etc.

Before we start implementing the interface to support advanced search functionalities, we study the various advanced search techniques followed by many standard websites and search engines.

6.2 Advanced Search Techniques

In this section, we study the various techniques [22 & 23] available to help the user to find the most appropriate search results that he/she is looking for. These techniques narrow down the search results by obeying certain criteria given by the user. Hence, the techniques below are usually the various criteria used for narrowing the search.

1. Case Sensitive Search

If a user wants to search for a keyword or phrase and gives particular importance to the case, then this advanced search technique is very useful.

E.g.) Keywords: “United States of America elections” should return results with initial capitalized words “United”, “States” and “America” and small case for “elections”

2. Phrase Search

The usage of double quotes enclosing a set of words (phrase) indicates to the search engine that the result should contain the exact phrase.

E.g.) Keywords: “class WelcomeServlet extends HttpServlet” should return results with the same phrase (same words, same order)

3. Language specific search

This helps to search for resources written in specific languages e.g. English, Italian, and German etc.

4. Truncation search

This strategy searches for various representation of the given keyword. The search engine returns any resource that contains plural, noun form, verb form etc of the given keyword or words in the phrase.

E.g.) Keyword: “Studies” would return resources with “study”, “studying” etc.

5. Logical operators

The use of logical operators AND, OR, NOT is very common in most of the search engines. This strategy is very helpful for users to narrow the search results. In most search engines these operators are not named as AND, OR, because these are technical terms or mathematical operators that common user may or may not understand. These options are represented by options like

“All words” → AND “Any words”, “At least one word” → OR E.g.) Keywords: “MPEG format audio file” searched with “All words” (AND) option would return resources containing all these words. The words may or may not appear in the exact order. Thus it differs from Phrase search.

Same keywords with “Any words” OR option would return resources with any one word. Thus the number of results returned by using OR option would be more than that using AND option.

6. +Requires and –Excludes search

Usually search engines avoid the stop words like “the”, “I” etc. If the user wants to include them in the search, the keyword is preceded with ‘+’ symbol. Similarly ‘-’ symbol is preceded before a keyword to exclude those words from

7. Proximity search

This technique searches for words appearing in a document within a certain distance or just separated by ‘n’ words. There could also be searches restricting the order of presence of the words.

E.g.) Keyword: “class HelloWorld extends HttpServlet” may have a proximity constraint that the word “class” and “extends” should be only separated by one word in between.

8. Date range search

Date ranges could be used for narrowing the search results. The user might be interested in resources published only after or before or within a certain period. Most of the search engines, especially job search engines come with this advanced search feature in order to provide the user with the recently posted jobs etc. But the implementation of this strategy differs widely across search engines. Few of them offer combo box or radio buttons to specify certain periods like “Within 1 week”, “Within 1 month” etc. Others might offer to search by specific date.

E.g.) Keyword: “Java developer” Posted within: “1 week” would return all jobs for Java developers posted within the past week.

9. Feature selection or Searching specific part of documents

It becomes quite important to search only a specific part of the document for the presence of a keyword. It is very useful in searching the emails for the presence of a word or phrase in the “Subject” or “From” or “To” etc. In documents we could search within the titles or specific names within the reference section etc.

10. URL Search

This option provides the users to search only for URLs.

11. Specific document type (format) or specific media search

Search engines offer this option to restrict the search only within specific file formats or media. Users could search within only PDF, PPT, text files etc. The search could also be done only for audio files, video files, images and other multimedia data.

12. Thesaurus search or Similarity search

This is an advanced feature used by many search engines to search for keywords and also other words that have similar meaning. Thesaurus contains a collection of similar meaning words grouped together. This thesaurus could be used by search engines to substitute the keywords with similar meaning words and thus produce better results.

Advanced Search Techniques used in current Search Engines

Features	Google	Yahoo	AltaVista
Phrase search	Yes – Using double quotes	Yes – Using double quotes	Yes – Using double quotes
Language specific search	Yes	Yes	Yes
Logical operators	Yes	Yes	Yes
Date range search	Yes – But not specific dates.	Yes – But not specific dates.	Yes – Also supports specific date search
Feature selection	Yes – title, URL, text, link	Yes	Yes
URL search	Yes – Domain specific also supported	Yes – Domain specific also supported	Yes – Domain specific also supported
File format specific search	Yes – PDF, Excel, Doc, HTML etc.	Yes – PDF, Excel, Doc, HTML etc.	Yes – PDF, Excel, Doc, HTML etc.

Table 6.1 Comparison of Advanced search techniques used in Google, Yahoo and AltaVista

6.3 Advanced Search Techniques in NEEDLE

1. Language specific

Search for e-learning resources documents, slides, audio and video published in Italian or English.

2. Logical operators

AND OR NOT options could be added to search for various combinations of keywords.

3. Date range search

Search for slides, lectures posted during a particular period.

4. Media specific search

NEEDLE system stores e-learning data as video lectures, audio, PowerPoint presentation slides. Therefore the user could opt to search only in some of these media.

5. Proximity search

This technique would be very helpful to search for learning materials that might contain source code etc.

6. Case sensitive search

This is also useful when searching source code or even technical documents that might contain class names or method names with case sensitive terms.

Implementation

The Simple Search Screen in NEEDLE provides a hyperlink to the Advanced Search screen which is implemented as a separate JSP. The JSP screen has been developed with the help of Ajax technology to provide user-friendly and highly dynamic screen. More on Ajax technology is discussed in the next section.

The query keywords entered by the user along with other search selections like date range etc. are sent to the '*AdvancedSearchServlet*'. After processing the user data the query string is formed based on Lucene query syntax [24] and sent to the Lucene API for searching the indexes.

To search using logical operators (OR, AND, NOT), Lucene API provides simple query syntax to be followed,

- ✓ OR – It is default operator for Lucene search. So, it's not required to mention the 'OR' keyword.
- ✓ AND – The keyword 'AND' is used between the search keywords. E.g. "java" AND "technology" – This displays results with both these words.
- ✓ NOT – The keyword 'NOT' is used before the words to be avoided. E.g. "java" NOT "technology" – Displays results with the word "java" without the word "technology".

In order to provide case-sensitive search, it is required to create two separate Lucene indexes i.e. one case-sensitive index and another lower-case index.

To perform date range search and language-specific search, some of the NEEDLE database table fields are used.

Address <http://localhost:8080/needle/advancedSearch.jsp>

Links [Customize Links](#) [Free Hotmail](#) [My Yahoo!](#) [Windows](#) [Windows Marketplace](#) [Windows Media](#) [Yahoo!](#) [Yahoo! Bookmarks](#) [Yahoo! Mail](#)

Google [G](#) [Go](#) [Orkut](#) [Bookmarks](#) [PageRank](#) [109 blocked](#) [Check](#) [AutoLink](#) [AutoFill](#) [Send to](#)

[Y!](#) [Pencil](#) [Magnifying Glass](#) [Y!](#)

needle

Next Generation Digital Library

Any of the words

All of the words

Exact Phrase

Without the words

Case Sensitive ☒

Search words Within words

Search in study resources [Presentation slides](#) Search words in [Title](#)

Languages ☐ Italian ☒ English

Date range [Between](#) From To

[Search](#)

Fig. 6.1 Screen Shot of NEEDLE Advanced Search Screen

6.4 Ajax

6.4.1 Overview

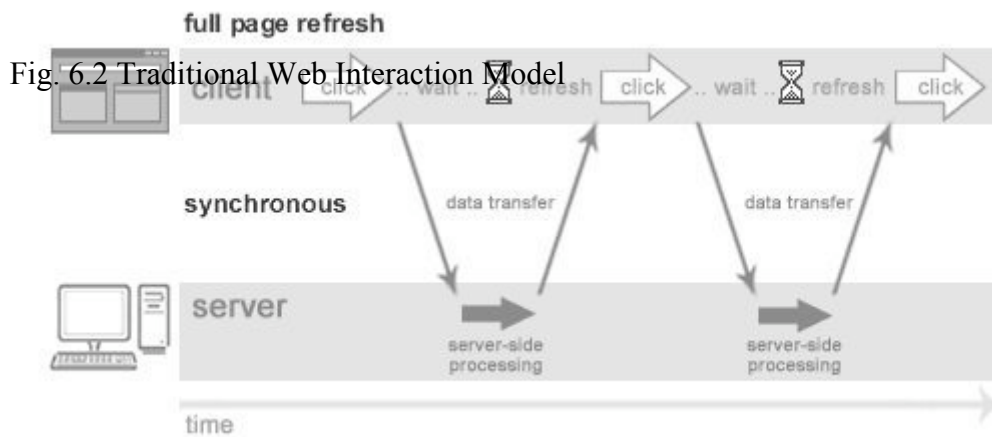
Recently, there has been development in creating high performance dynamic web applications. Applications like GMail, Google Maps etc. has been known for using a new technology called the Ajax - Asynchronous JavaScript Technology [25] and XML to provide such dynamic web pages. There is close similarity in dynamic behavior between these new breed of web applications using Ajax and that of traditional desktop applications. The users can interact with these web applications just like a desktop application. They do not depend on any browser specific plug-ins or any installations.

So far the web applications have been a set of HTML pages that is made dynamic by reloading the whole content (page) even for a change in a small portion of the page. Scripting technologies like JavaScript programming language and cascading style sheets (CSS) have been quite successful and they can be used effectively to create very dynamic web applications that will work on all of the major browsers. Still, the dynamic behavior of web applications is not as rich and interactive as desktop applications. Ajax helps to bridge this gap.

6.4.2 Classical Web Interaction Model

The traditional web applications were originally designed for browsing HTML documents. In this regard, the user interaction model for web application was a "click, wait, and refresh" model and a "synchronous request/response" communication model [25 & 26]. It is explained as follows:

1. **"Click, wait, and refresh" user interaction model:** When a user clicks or submits a HTML page, the browser discards the current HTML page and sends an HTTP request back to a web server. The server processes the request some and then returns an HTML page to the browser. The browser refreshes the screen and displays the new HTML page.
2. **"Synchronous request/response" communication model:** The communication between the browser and the web server is always a synchronous communication. The browser initiates requests, whereas the web server just responds to the browser requests. The "request/response" cycle is synchronous, during which the user has to wait.



The above mentioned two characteristics of traditional web have many disadvantages in the context of software applications: long waiting times for user due to "click, wait, and refresh", slow performance, server overload and bandwidth consumption due to redundant page refreshes, and lack communication support for server initiated updates. Thus it results in slow, unreliable, low performance and inefficient web applications.

6.4.3 Ajax Interaction Model

These two basic behaviors of "click, wait, and refresh" and "Synchronous request/response" must be altered to produce high performance, more interactive and efficient web applications. This has been achieved in the Ajax interaction model [25 & 26], which is explained as follows:

1. **"Partial screen update"** – In this type of user interaction, only a part of the HTML page that contains new information is refreshed and updated and the rest of the user interface remains displayed without interruption. This "partial screen update" interaction model thus enables continuous operation.
2. **"Asynchronous communication"** – The asynchronous request/response communication model used in Ajax decouples user interaction from server interaction. As a result, the user can continue to use the application while the client program requests information from the server in the background. When new information arrives, only the corresponding user interface portion is updated.

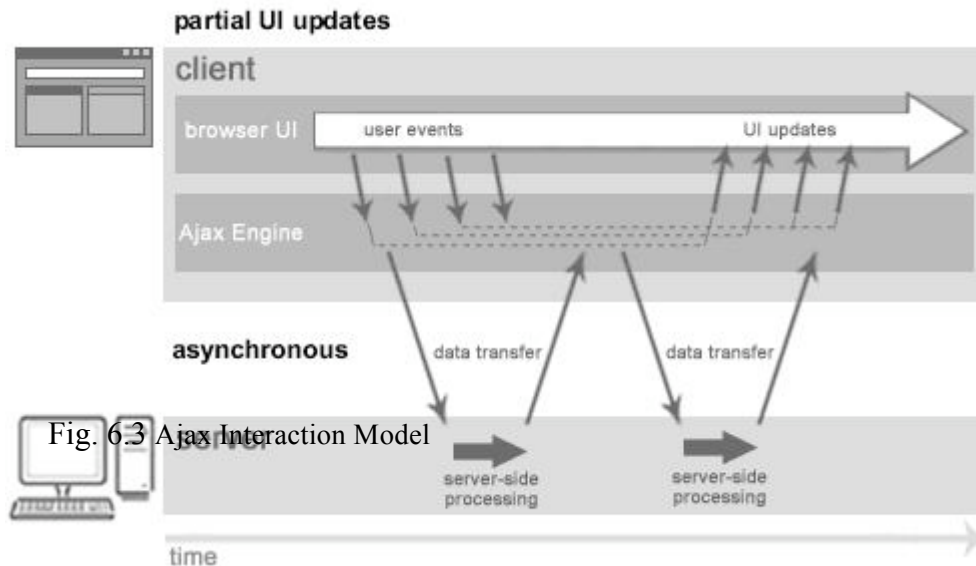


Fig. 6.3 Ajax Interaction Model

6.4.4 JavaScript to Ajax Technology

The term Asynchronous JavaScript Technology and XML (Ajax) has been used recently to define the interaction of JavaScript with the server in a slightly different fashion. It is possible to make asynchronous calls using JavaScript technology from HTML page to the server and fetch content in different efficient formats like XML documents, plain text, HTML, JavaScript Object Notation (JSON) etc. The retrieved content may then be used to update the Document Object Model (DOM) of the HTML page. This type of interaction model has been defined as the Ajax technology.

Ajax technology has been around for quite sometime. It was previously known as web remoting or remote scripting and has been available to Internet Explorer developers on Windows. Some developers have simulated this type of Ajax interaction model using a combination of plug-ins, Java applets and hidden frames. The key difference in today's technology is the support for the *XMLHttpRequest* [27] object in the JavaScript runtimes of the web browsers. This support for *XMLHttpRequest* object in JavaScript has been found extremely useful for the Ajax type interaction. There are still minor differences among current browsers such as Mozilla Firefox, IE etc with the support for JavaScript. But all these are manageable. There are many JavaScript libraries available to bridge the gap and to provide a standardized programming language. Dojo, for example, is addressing accessibility, internationalization, and advanced graphics across browsers – which were all great barriers till now. More libraries like Scriptaculous, Prototype, and Yahoo User Interface Library were introduced to solve different challenges. And they have proved to be quite successful.

Ajax technology is based on event generation and suitable action. Using Ajax, one can achieve a clear separation of presentation logic from the data content. Usually, the server generates HTML documents for every client event resulting in a call to the server. The clients would then reload and render the complete HTML page for each response. But, this would result in huge amount of data transfer and delay in rendering the whole page with the new content. Instead, new rich web applications focus on a client fetching an HTML document that remains as a template and then the content is filled and updated based on client events using XML data retrieved from a server-side component.

6.4.5 Scenarios showing Ajax interactions

The following are some of the usage scenarios [27] of Ajax,

- **Auto completion:** There could be some fields which need to be completed automatically as a user types in a few characters. E.g. an email address, name, or city name etc.
- **Real-time form data validation:** Some fields like user names, serial numbers needs to be validated at the server-side even before the user submits a form. As soon as the user types in the data, this small chunk is sent to the server and validated.
- **Advanced user interface controls and effects:** Controls such as trees, calendars, and progress bars etc. allow for better user interaction and interaction with HTML pages, generally without requiring the user to reload the page.
- **Refreshing data and server push:** There are some web applications which require to be updated frequently or at regular intervals like for e.g. stock process, scoreboards etc. In these cases, HTML pages may poll data from a server for up-to-date data. A client may use Ajax techniques to get a set of current data without reloading a full page.
- **Partial submit:** There could be scenarios to just submit a portion of an HTML page without requiring a full page refresh.
- **Mashups:** A web application could receive data from more than one external source and then merge to present it on a single HTML page. For example, it is

possible to mix data from a third-party application such as Google Maps with our own application.

- **Page as an application:** Ajax can be used to create single-page applications that are similar to and function like a desktop application.

Chapter 7:

Service Oriented Architecture for

NEEDLE Search Library

7 Service Oriented Architecture for NEEDLE Search Library

7.1 Service Oriented Architecture - Evolution

The earlier distributed computing systems like COBRA, Smalltalk, and Java RMI etc had very tight coupling between the various components in the system for the purpose of ubiquitous e- business over the internet. These architectures require too much compliance and shared agreement among different business vendors [33]. This supports for low overhead and open Business -2-Business e-business.

The above situation is manageable to a certain extent through skills and number of people. As the size and rate of business change increases, this weakness becomes a major issue. Any significant change in any one of these aspects will cause a great damage to the system like unavailable or unresponsive web sites, lack to cope up with new market trends and services and inability to shift to new business opportunities.

The major weakness of this conventional approach is the communication link between both the partners needs to be implemented using the same distributed object model (Java/RMI, DCOM or CORBA/IIOP). There is problem of making these protocols to work beyond firewalls or proxy servers. For e.g. most of the firewalls are configured to allow Hypertext Transfer Protocol (HTTP) to pass across, but not IIOP [28 & 33]. These Monolithic systems are very sensitive to changes as any modification to the output of one of the subsystems will result in the break down of the whole system. Any attempts to provide a new implementation of a subsystem will also often cause the old system to break down since the implementations are tightly bound to each other.

Hence there arises the need to replace the current models of application design with a more flexible architecture, yielding systems that are more ready for changes. The current trend of application architecture shows a shift from tightly coupled systems towards loosely coupled systems. Systems built with these types of architectures are more likely to govern the next generation of enterprise applications. The key feature of success of these systems is flexibility.

Web services architecture [29] just consists of loosely coupled services and has evolved from object oriented systems which consisted of objects and components. The services in the system encapsulate behavior and provide a messaging API to other interacting components on the network.

These applications are based on compositions of services which are dynamically discovered at runtime called as just-in-time integration of services or pre-fixed during design.

At runtime, application execution is a matter of translating the requestor requirements into input for a discovery mechanism and locating a right vendor capable of providing the appropriate service and invoking the services. Web Service architecture describes many techniques for the implementation of web services.

7.2 Web Service Architecture (WSA)

Web Service Architecture (WSA) [29] is meant to provide a standardized mechanism for interoperating between heterogeneous software applications, running on different types of platforms and frameworks. It provides a model for understanding Web services and shows the relationships between different specifications and technologies that are involved. This promotes interoperability between the different compatible protocols.

The WSA does not provide any details about the implementation of Web services, and there is no restriction on how these services are to be combined. It provides an interoperability framework for the future evolution of Web services standards. That framework must be able to work on both pure SOAP-RPC and HTTP.

7.3 Interoperability Architecture

While talking about interoperable applications, it is important to ensure certain concepts, constraints and relationships in the design process. Such issues are identified by this Interoperability architecture [28 & 29]

The major goals of this architecture is to provide [29]

- interoperable Web services
- integration with the World Wide Web
- reliability and security of Web services
- scalability and extensibility of Web services

This architecture does not include

- any specific programming language
- any specific Web services implementation
- any Web services construction details
- any specific details about messages, message discovery and choreography etc.

Benefits of SOA

1. SOA Infrastructure supports heterogeneous environment across operating systems, different applications etc.
2. Reuse and composition: This is helpful for creating new business processes quickly and reliably. Since existing business applications are bound to changes with the business change, reconstruction from the scratch is not possible. Hence SOA applications are built to respond to changes quickly.
3. It supports a new method of interactions between customers, partners and suppliers.
4. The facility to alter existing business processes or applications based on service aggregation. SOA's loosely coupled nature allows the easy plug in of new services and also upgrades the existing services to meet the new business requirements.
5. It also helps in exposing the existing business applications e.g. legacy applications as services, hence safeguards the existing IT infrastructure investments.
6. It has the ability to incrementally build the system. This is true for SOA-based integration.

Web Service Distributed Management

7.4 SOA Infrastructure

Web Service Orchestration

Web Service Coordination

Web Service Transactions

Web Service Security

Web Service Policy

Web Service Reliable Messaging

Figure 7.1 illustrates the SOA Infrastructure

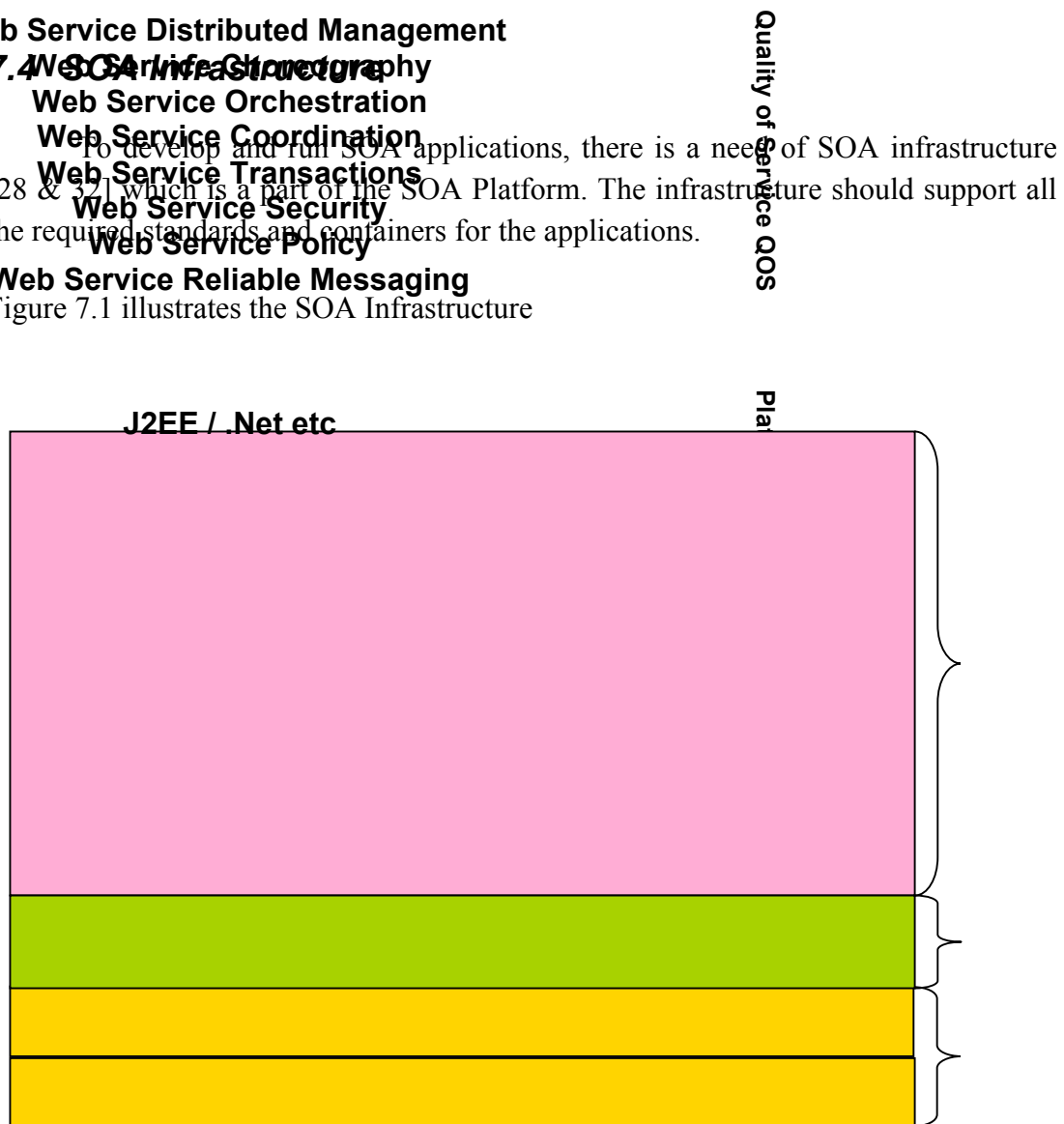


Fig. 7.1 SOA Infrastructure

SOAP, WSDL, UDDI

These are the fundamental blocks of the SOA infrastructure. SOAP is used as a transport layer to send messages between service providers and consumers. SOAP is the default mechanism supported for web services, also other alternative technologies can be used for binding services. WSDL is used to describe the service. UDDI is used for registering and for services lookup. Any service consumer can search for the service using UDDI, get the WSDL of the corresponding service and invoke the service using SOAP.

WS-I Basic Profile

This is used for Service testing and interoperability. Service providers before consuming the service can test for interoperability across heterogeneous platforms and technologies by using the basic profile test suites.

Security WS-Security	Reliability WS-Reliable Messaging	Transactional WS-coordination	Description
J2EE, .Net Messaging SOAP, WS-Addressing, WS-Notification Most SOA Applications are developed in J2EE and .Net platforms, but SOA also supports other platforms. J2EE platforms not only provide the framework for the developers to build SOA applications, but also includes great infrastructure to support reliability, scalability and better performance. Java API for XML binding (JAXB) is used for mapping XML documents to java classes and Java API for XML-based Remote Procedure Call (XML-RPC) is used for invoking remote services in J2EE 1.4 for the development and deployment of Web services. These are portable across standard J2EE containers and also interoperable with services across different platforms such as .Net.			WSDL, UDDI, WS-Policy
XML Transports Example: HTTP, TCP/IP			

7.4.1 SOAP

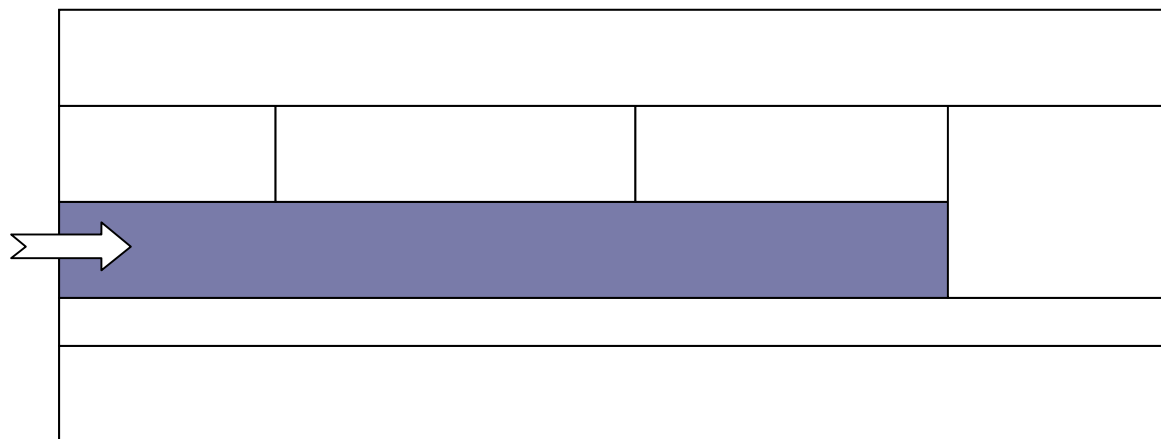


Fig. 7.2 Web Services Stack

SOAP [29 & 32] is an XML-based light weight communication protocol for exchanging messages between computers regardless of their operating systems, programming environment, or object model framework in a distributed environment enabling remote method invocation.

The W3C introduced the specification for SOAP in 1999. The current W3C recommendation is Version 1.2.

SOAP covers the following four main areas:

1. Message Format:

It is used for one-way communication defining how a message can be transmitted as an XML document.

2. Description:

It defines the mechanism for transporting a SOAP message using different protocols like HTTP, SMTP (for e-mail based interaction) etc.

3. Set of Rules:

It defines the rules for processing a SOAP message. It also specifies the intended recipient for specific parts of the message and how to react in case the content is not understood.

4. Set of Conventions:

It describes how to convert an RPC call into a SOAP message and back and also how to implement the RPC method of interaction.

When the client makes an RPC call, it is translated into a SOAP message, forwarded and again it is turned into an RPC call at the server. The reply from the server is converted into a SOAP message and sent back to the client. When it is passed on to the client it is again turned into an RPC call.

Basic SOAP Mechanism [32]

- a) Messages are encapsulated into XML document
- b) The XML document with SOAP message is converted into an HTTP request.
- c) RPC calls are converted into SOAP message
- d) Rules to process the SOAP message

SOAP Messages

Header block

SOAP is based on asynchronous message exchanges

SOAP Body

Body block

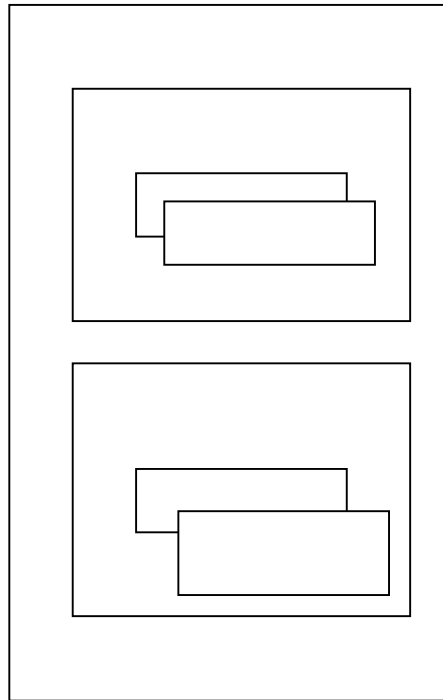


Fig. 7.3 SOAP Message Format

A message has two main parts namely, header and body. Header is optional whereas body is mandatory. Header contains the infrastructure level data and the body contains application level data.

SOAP Header element contains blocks of information about how to process the message. This provides the way to convey the information in the soap message other than payload.

SOAP Body element is the mandatory part of the SOAP envelope. This carries the main information or payload between the partners.

Body is where the method call information and its related arguments are encoded. It is where the response to method call is placed, and where error information can be stored.

Application specific data

It is the actual information that is exchanged with a web service. This can be a XML data or parameters for a method call.

Fault message

It is used only when some error occurs. This error can be due to processing error or an improperly structured message. SOAP can either have application specific data or fault message but not both.

SOAP communication Model [28 & 32]

Two possible communication styles are:

- ✓ Remote Procedure Call (RPC)
- ✓ Document or Message

The four different forms of communication are :

- ✓ RPC/Literal
- ✓ Document/Literal
- ✓ RPC/Encoded
- ✓ Document/Encoded

Remote procedure call (RPC)

In this type the web services appear as remote object to client application. This type is used when the client invoking a web service needs an immediate response (synchronous) and both the client and the web service communicate in a conversational way. This is process oriented rather than data oriented.

Document-style

When a client invokes a document style web service the client request is sent as an entire document rather than individual parameters. The web service may or may not return a response. This is an asynchronous type, so the web service can continue with its work without waiting for the response. This is data oriented rather than process oriented.

Remote procedure call (RPC) Vs Document-style

Document style does not support serialization / de-serialization of messages, whereas it assumes that SOAP messages are well formed XML documents. Since Document style is asynchronous, it promotes loose coupling between the client and the service provider. When client does not need any immediate response then document style can be used.

Compositional Collaboration BPEL, WS-CDL

Security
WS-Security

Reliability
WS-Reliable Messaging
7.4.2 WSDL
Messaging
SOAP, WS-Addressing, WS-Notification

Transactional
WS-coordination

Description

WSDL,
UDDI,
WS-Policy

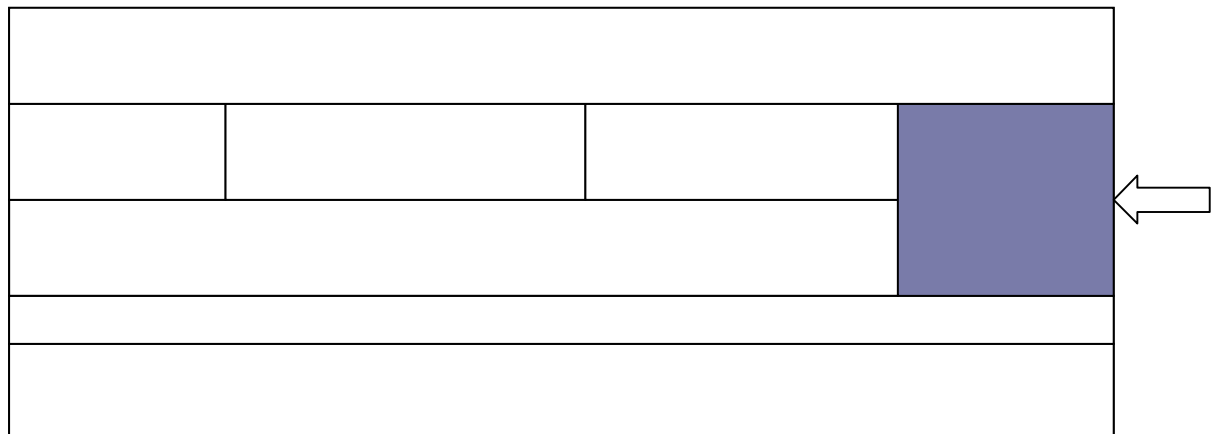


Fig. 7.4 Web Services Stack

To make use of SOAP with any web service it would need some documentation that explains the structure of the soap message that is being exchanged. It requires the operations that are being exposed along with the parameters that are required in machine readable format. It is required to know which protocol needs to be used like HTTP or SMTP etc. It is also required to know the internet address of the required web service. Therefore a standard description language is required to solve the problems.

Web Service Description language (WSDL) [32] is the service representation language that is used to describe the necessary details that are exposed by the web services. Hence WSDL is used for accessing any required web services.

WSDL is very helpful such that the service requestor or the service provider need not know about each others infrastructure details, technical implementation details or object framework etc.

WSDL is a platform and language independent and it is mainly used to describe the exposed web services.

WSDL mainly describes:

1. **What a Web Service Does** – This describes the operations supported by the service provider.
2. **Where the Web Service is** – This describes the protocol details, URL address etc.

3. **How to Invoke the Web Service** – This explains the details of the data format, parameters and the necessary details required to access the service operations

WSDL documents contains two distinct sections

Service Interface Definition:

This describes the interface structure of the web service. It contains the operations supported, data types used, the parameters required to invoke the web service.

Service Implementation Part:

This relates the interface of the service to the network address and describes about the protocol and the data structures used. Any client service can bind to such an implementation and invoke the corresponding web service.

Service Interface part consists of the following WSDL elements [32]

Port Type:

This element describes the operations supported by the web service, messaging mode and payload details. It does not specify the protocol used or the address of the web service. It also specifies all the operations supported by the web service. Collection of operations at an end point is grouped as port types.

Operation:

This resembles the method signatures. This includes the name of the method, input and output parameters. It also defines the faults. Each Port type can have one or more operation elements.

Message:

This element describes the payload of the messages that are being sent out and received from a web service. It also describes the contents of the SOAP header block and fault details.

Types:

It contains all the data types that are present in the web service interface. It is similar to the data type in Java etc. It defines primitive data types like int, float etc, developers can also build complex data types from them in the messages.

Query for all
matching services

7.4.3 UDDI (Universal Description, Discovery, and Integration)

Service
Registry

UDDI [32 & 33] is a platform-independent, XML based registry for all the service providers to list themselves on the internet. It is an open industry initiative, given by OASIS to enable the service providers to publish their services and discover each other services.

Request for selected
service info

Service info of
selected services

Publish

The problem of how to locate the web service from a large collection of services is solved by using service discovery mechanism.

Search

Service discovery is the process of locating the providers of the web services and getting the web services descriptions that have been already published. On the completion of the discovery process the client application must know the exact location of the web service i.e. URI and the process of how to interface with it.

Invoke

Service
Provider

Invocation response

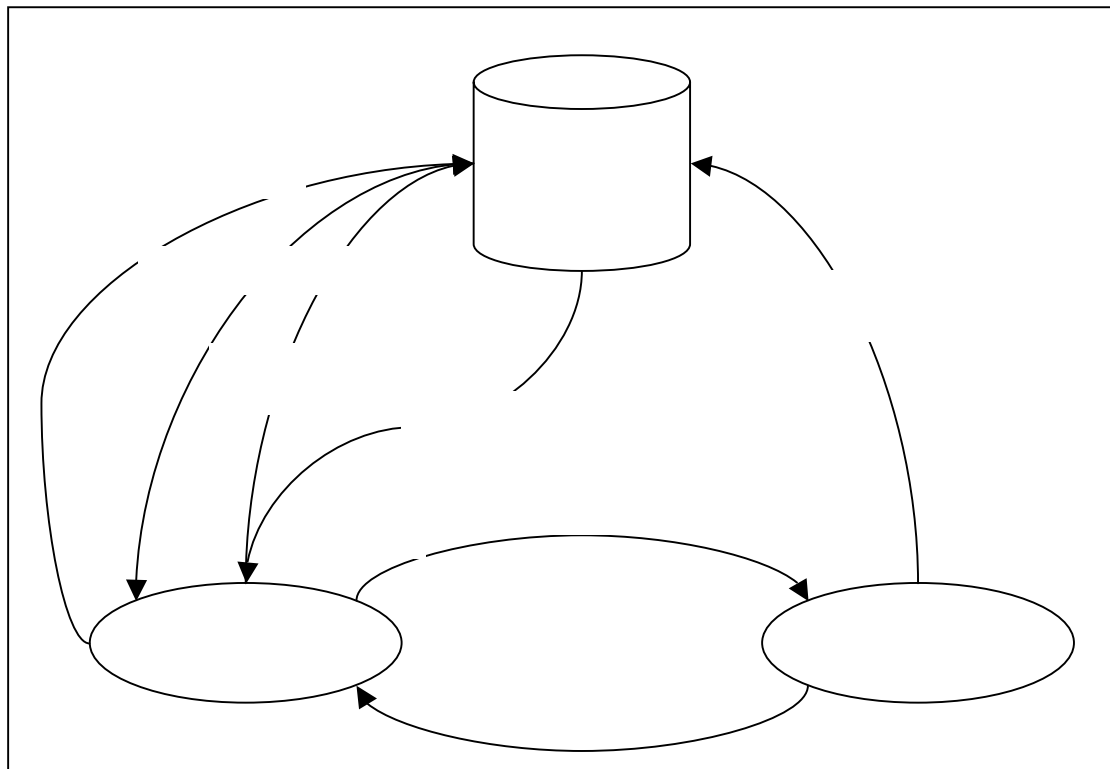


Fig. 7.5 UDDI Service Registry

Two Types of Service Discovery

- Static
- Dynamic

Static Discovery:

It is similar to OOP. Here the service implementation details are bound during the design time. The Web service is retrieved from the service registry.

Dynamic Discovery:

This is an adaptive SOA pattern. Here the service implementation details are bound during run time. Here one or more services are retrieved from the service registry that matches the interface definition; the client application chooses the most suitable service, binds to it and invokes it.

7.5 Apache Axis2

Apache Axis2 [30 & 31] is an open source platform that provides the basic for the development of web services and it supports both SOAP (Simple Object Access Protocol) and REST (Representational State Transfer).

Axis2 Features

1. *Speed* - Axis2 uses its own object model and StAX parsing to attain greater speed than compared to previous Apache Axis versions.
2. *AXIOM* - Axis2's light-weight object model called AXIOM is very efficient and has high performance message processing.
3. *Hot Deployment* – New services can be deployed to the system without stopping the server. Once the Axis archive file is placed in the services directory, it is automatically deployed.
4. *WSDL support* - Axis2 supports the WSDL version 1.1 and 2.0, which allows building stubs for accessing remote service. It automatically creates the machine readable description for the deployed services.
5. It has high stability and flexibility of web services.

7.5.1 Creating Web Services Using Axis2

A Web service is represented by a service provider side code called as Skeleton and client side code called as Stub. These stubs and skeletons are responsible for marshalling and un-marshalling service requests. First step in creating web service is creating the service provider side skeleton for the service.

Web service can be created using 2 approaches

- **bottom-up approach or Code first approach**
- **top-down approach or Contract first approach**

In bottom-up approach, first the java classes are written and then the service descriptor WSDL is generated from it. If the java classes are already available then the bottom-up approach can be done.

In top-down approach, WSDL is created first and then Java classes are generated from it.

Axis2 supports both synchronous and asynchronous services. Once the web services are created then it has to be deployed and tested with a client. Axis2 libraries have to be installed in the library directory of the application classpath.

Steps in creating a web service using Bottom-up approach:

1. As the first step, make sure Axis2 libraries are included in the application development project.
2. Write the service implementation class (like a simple java class) which contains methods that perform some task. These methods will be invoked by external applications.
3. Write a service descriptor (services.xml).
4. Create a service archive file.
5. Deploy the archive file in the Axis2 directory placed in a Tomcat server.

Writing a Service Descriptor file (Services.xml)

To create a service archive file for deployment services.xml file should be first written. Services.xml file is a deployment descriptor file for the service that is to be deployed. Deployment descriptor explains the deployment module, how to configure and deploy the service.

Services.xml file should contain the following details

1. Fully qualified class name of the service implementation class
2. Message receiver that is going to be used

Axis 2 has a set of built in message receivers they are:

1. Raw XML- can handle XML-in and XML out methods
2. RPC message Receivers – can handle java beans, simple java types and XML.

Services.xml file looks like:

```
<service>
  <description>
    Description about the Service
  </description>
  <parameter name="ServiceClass">SampleService</parameter>
  <operation name="helloWorld">
    <messageReceiver
      class="org.apache.axis2.rpc.receivers.RPCMessageReceiver"/>
  </operation>
</service>
```

Creating a service archive file

The service archive file (.aar) is created with the compiled java code of the service implementation class and the services .xml. To deploy the service archive file (.aar), it is placed inside the services directory in the Axis2 which in turn is present

inside Tomcat server. The deployed services can be accessed by the axis2 web administration console.

Service group and single service

Service group is used to deploy more than one service together in a single service archive file (.aar). Hence there can be multiple service implementation files and a single services.xml file for them. The only change in the services.xml file is the root element which is changed to service group instead of service.

The services.xml for 2 services Service A and Service B will look like the following

```
<serviceGroup>
  <service name=" Service A ">
    .....
  </service>
  <service name=" Service B ">
    .....
  </service>
</serviceGroup>
```

Steps in creating a web service using Top- down approach:

In this approach the web service is created from the WSDL. Here both the client and the service provider are provided with the WSDL and they start building the web service form that.

The following steps are done

1. The Service code skeleton is developed
2. Service skeleton is written according to the business agreement
3. Run the Ant build file
4. Deploy service archive file into the application server e.g. tomcat where Axis2 is running

7.5.2 Generating Service code from WSDL

Axis2 has a set of tools and IDE plug-ins for the code generation [31] from given WSDL. Any data binding can be chosen from the available like xmlbeans, adb etc. After the server code is generated it generates the services.xml, ant build file, message receivers and service skeleton class.

Five ways to create a web service

1. Plain Old Java Objects (POJO),
2. Axiom's OMElement,

XML Stream

3. Axis2 Data binding Framework (ADB)
4. XMLBeans,
5. JiBX.

Axiom

Of the above the most used approaches are POJO and Axiom.

StAX Writer API**POJO Approach:**

The Plain Old Java Object (POJO) is a simple and quick way to make most of the already existing Java classes available as Web services. One main disadvantage of POJO based Web services is the lack of initialization support, that values needs to be initialized before the web service is used.

AXIOM Approach:

AXIOM depends on StAX API for input and output data. The important part of the AXIOM Approach is deferred building which other approaches does not support. This will build the model only whenever required by the application. AXIOM also has the ability of caching StAX events with or without building the model. Hence it provides better performance compared to the POJO approach.

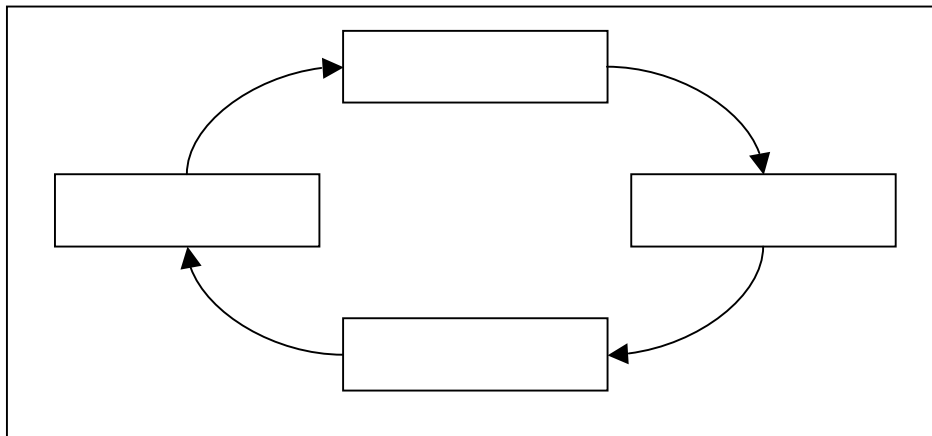
AXIOM Architecture

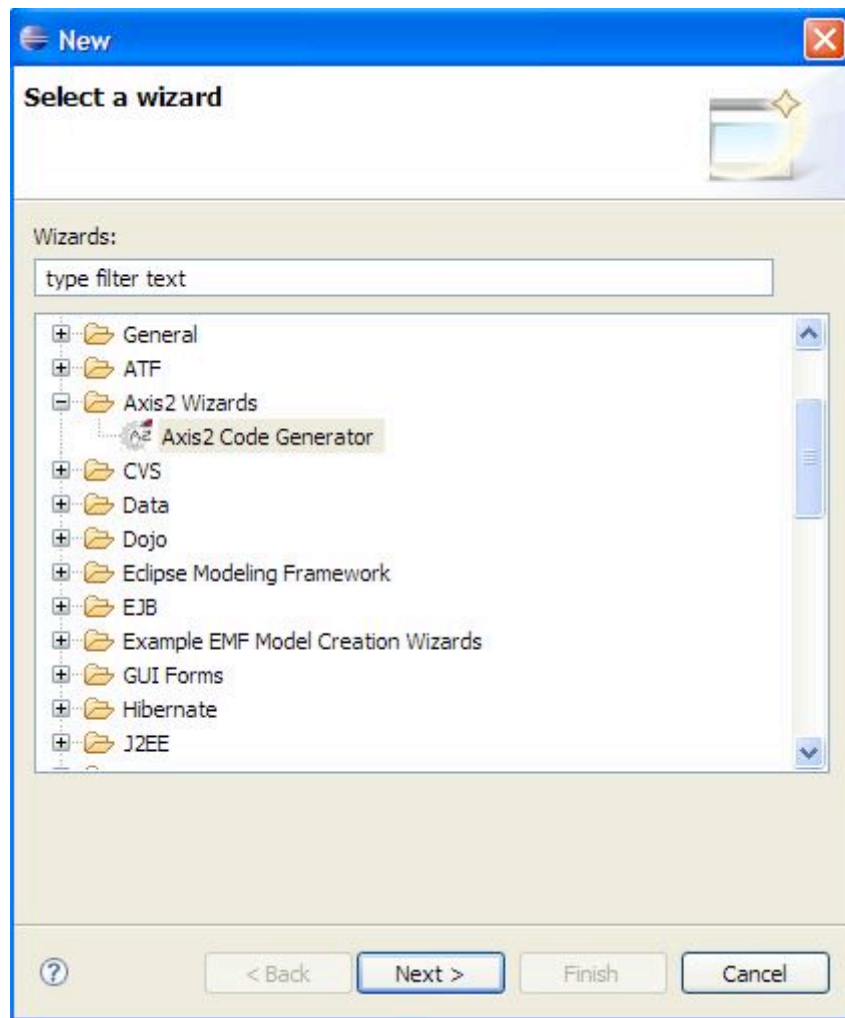
Fig. 7.6 AXIOM Architecture

7.5.3 Axis2 code generator Plug-in

Axis2 code generator plug-in for Eclipse [31] is used for generating Java class from a WSDL file and generating a WSDL file from a Java class.

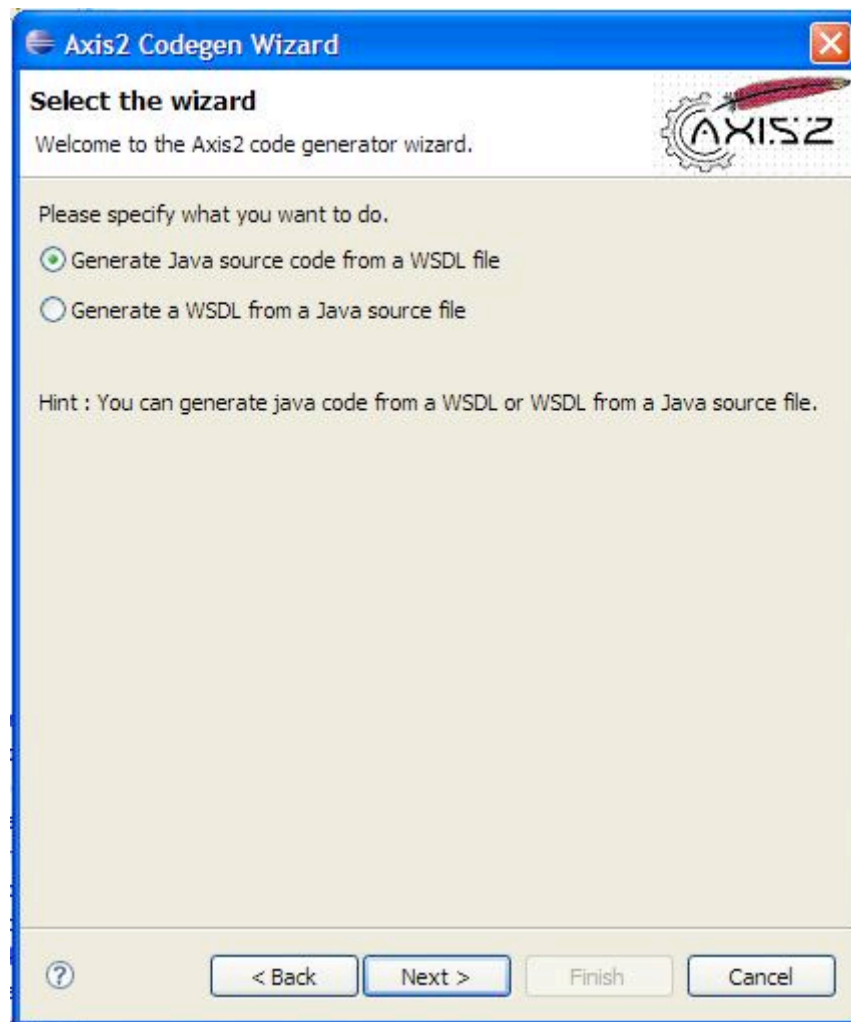
Steps for generating a Java Class from the given WSDL [31]

1. Download the Axis2 code generator plug in for eclipse
2. After the successful installation of the plug in, File→ New→ other shows the following wizard



Screen-1

3. Select “Axis2 Code Generator” and click the next button. The following screen appears



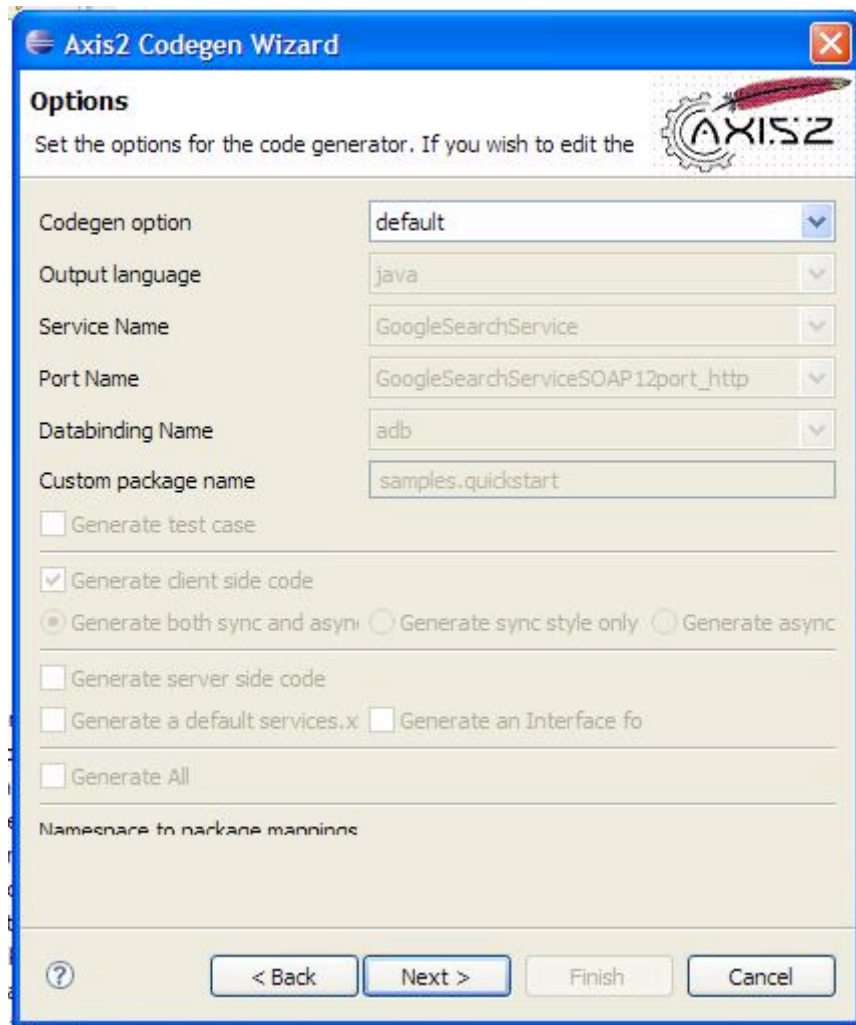
Screen-2

4. Select the first option ("Generate Java Source code from a WSDL file") for generating the java source code from the WSDL file and click on next , the following screen appears



Screen-3

5. Select the location of the corresponding WSDL using the browse button and click Next, the following screen appears



The image shows the 'Axis2 Codegen Wizard' dialog box, specifically the 'Options' tab. The title bar reads 'Axis2 Codegen Wizard' with a close button. Below the title bar, the text 'Options' is followed by a description: 'Set the options for the code generator. If you wish to edit the'. To the right of this text is the Axis2 logo, which consists of a gear and a red pen. The main area of the dialog contains several configuration options: 'Codegen option' is a dropdown menu set to 'default'; 'Output language' is a dropdown menu set to 'java'; 'Service Name' is a dropdown menu set to 'GoogleSearchService'; 'Port Name' is a dropdown menu set to 'GoogleSearchServiceSOAP12port_http'; 'Databinding Name' is a dropdown menu set to 'adb'; 'Custom package name' is a text field containing 'samples.quickstart'. Below these fields are several checkboxes: 'Generate test case' (unchecked), 'Generate client side code' (checked), 'Generate both sync and async' (selected radio button), 'Generate sync style only' (unselected radio button), 'Generate async' (unselected radio button), 'Generate server side code' (unchecked), 'Generate a default services.x' (unchecked), 'Generate an Interface fo' (unchecked), and 'Generate All' (unchecked). At the bottom of the dialog, there is a question mark icon, a '< Back' button, a 'Next >' button, a 'Finish' button, and a 'Cancel' button.

Axis2 Codegen Wizard

Options

Set the options for the code generator. If you wish to edit the

Codegen option: default

Output language: java

Service Name: GoogleSearchService

Port Name: GoogleSearchServiceSOAP12port_http

Databinding Name: adb

Custom package name: samples.quickstart

☐ Generate test case

☒ Generate client side code

☒ Generate both sync and async ☐ Generate sync style only ☐ Generate async

☐ Generate server side code

☐ Generate a default services.x ☐ Generate an Interface fo

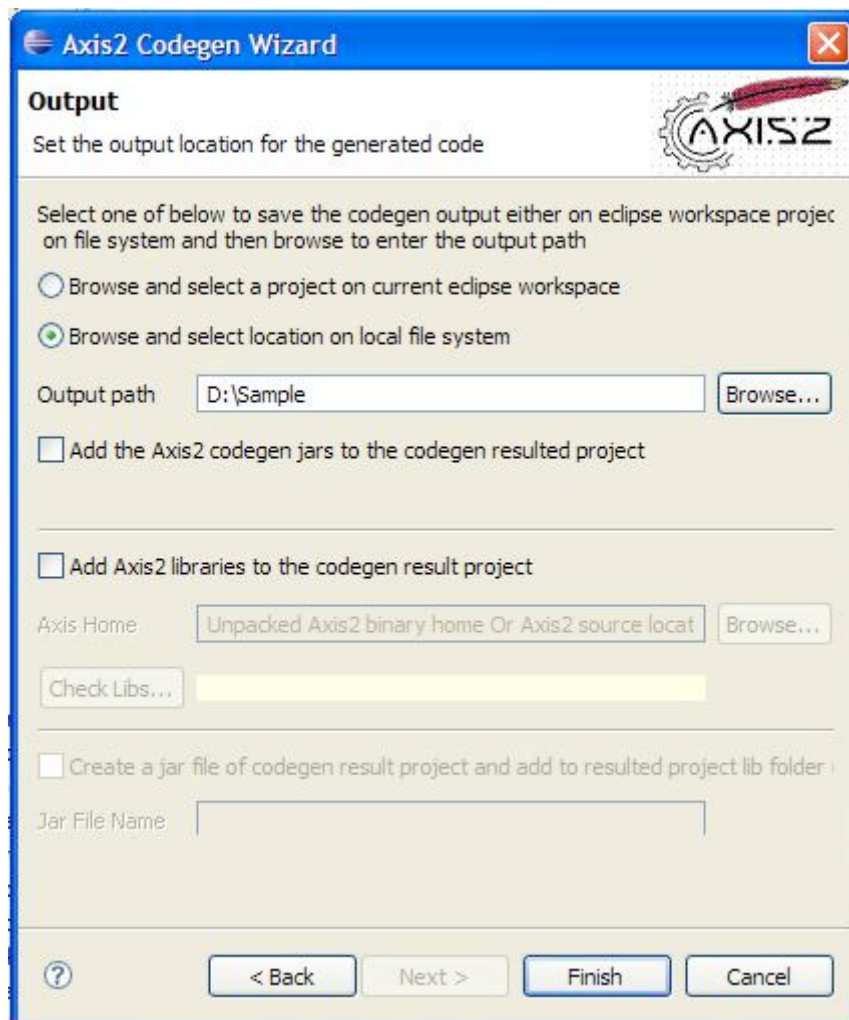
☐ Generate All

Namespace to namespace mappings

? < Back Next > Finish Cancel

Screen-4

6. Select the corresponding service name, package name, port name etc and click next. The following screen appears



Screen -5

7. Select the output file path using the "Browse" button. Select the corresponding location to store the output file. Verify the availability of the Axis2 libraries by clicking on the "Check Libs" button. Once everything is done click the "finish" button and the success message is displayed.
8. Finally, the java stub code is generated for the given WSDL in the given output location.

7.6 *NEEDLE Search Library*

7.6.1 Motivation

NEEDLE as a separate web-based application can be used to search e-Learning resources that has been pre-processed and uploaded into the NEEDLE backend. The simple and advanced search functionalities help to refine the queries and retrieve the desired results. However, from a user's perspective, the resources are limited in using just NEEDLE alone. It would be better to provide a broader domain of searching for the materials.

There are many commercial search engines like Google, Yahoo etc. and other educational, knowledge websites like "citeseer" etc. hosting articles, technical journals, e-books, sample programs etc. These are good resources to search for learning materials. The user interface for NEEDLE could also be used to extend the search in these websites to deliver more hits. This idea brings us to the concept of meta-search engines.

Meta-search engines have a single search interface for taking the user input query and perform search on multiple search engines. The search hits from all the sources are collected; formatted and presented in a sorted / ranked order (the ranking used is usually that of the search engine). Meta-search engines usually do not implement any search or ranking algorithm. They just give these responsibilities to the search engines.

Thus, the usage of such meta-search in NEEDLE helps the users to compare the results from various e-Learning sources. In order to achieve this meta-search functionality, we introduce the concept called '*Search Libraries*'. Apart from the general NEEDLE search, the users could configure a few external search engines of their interest and select them while performing a search. The query would be sent to the selected engines and the results would be presented. One such configured search engine is called a Search Library.

The search engines like Google, Yahoo etc. have introduced search APIs based on Service Oriented Architecture (SOA). This means, Google provides a Web Service for receiving a query from an external application, performs search and returns the result to the invoking application. Thus in our case, we can develop web services for each of the search engines i.e. Search Library and invoke them from NEEDLE business logic.

As a result of using SOA, NEEDLE system has entered a new generation of distributed computing, which enables it to interact with external applications

implemented on different technologies and platforms. SOA provides a common interface for interacting with other applications.

7.6.2 Requirements

We now study the requirements for developing the Search Libraries and the interaction of such components with NEEDLE system through SOA architecture.

The following are the functional requirements of a Search Library

1. The Search Library corresponding to a search engine or domain should be displayed as an image or icon in the Search JSP screen for the user to select it for searching.
2. The query entered by the user should be sent to all the selected Search Libraries and the results collected separately and presented to the user.
3. The search hit results from different search engines could have different formats. However, while integrating the results from all the search engines, it is required to present a consistent format of the results. Therefore, the search module should perform this conversion from search engine specific format to common format for presenting to the user.
4. User should be able to configure a new Search Library and integrate it with NEEDLE.
5. As the search APIs provided by the standard search engines like Google, Yahoo etc. are built as web services, the component used to call these should follow SOA.
6. There could be two types of Search Libraries
 - a. *Search Engine with Web service* – In this type, user can configure a new Search engine with a corresponding web service implementation for performing the search.
 - b. *Search Domain* – There are some useful websites which host huge volumes of materials, but do not provide a search interface. For such sites, we make use of standard search engines like Google, to search the particular website i.e. we restrict the search domain of Google. In this case, we need not implement a separate web service. Rather, we use the Google web service and provide the website (domain) as input.

7.6.3 Design

As discussed earlier, we are moving to SOA for the implementation of Search Library in NEEDLE. A brief discussion of SOA and related concepts was done in the previous section.

The Figure 7.7 illustrates the architecture design of the Search Library with NEEDLE system.

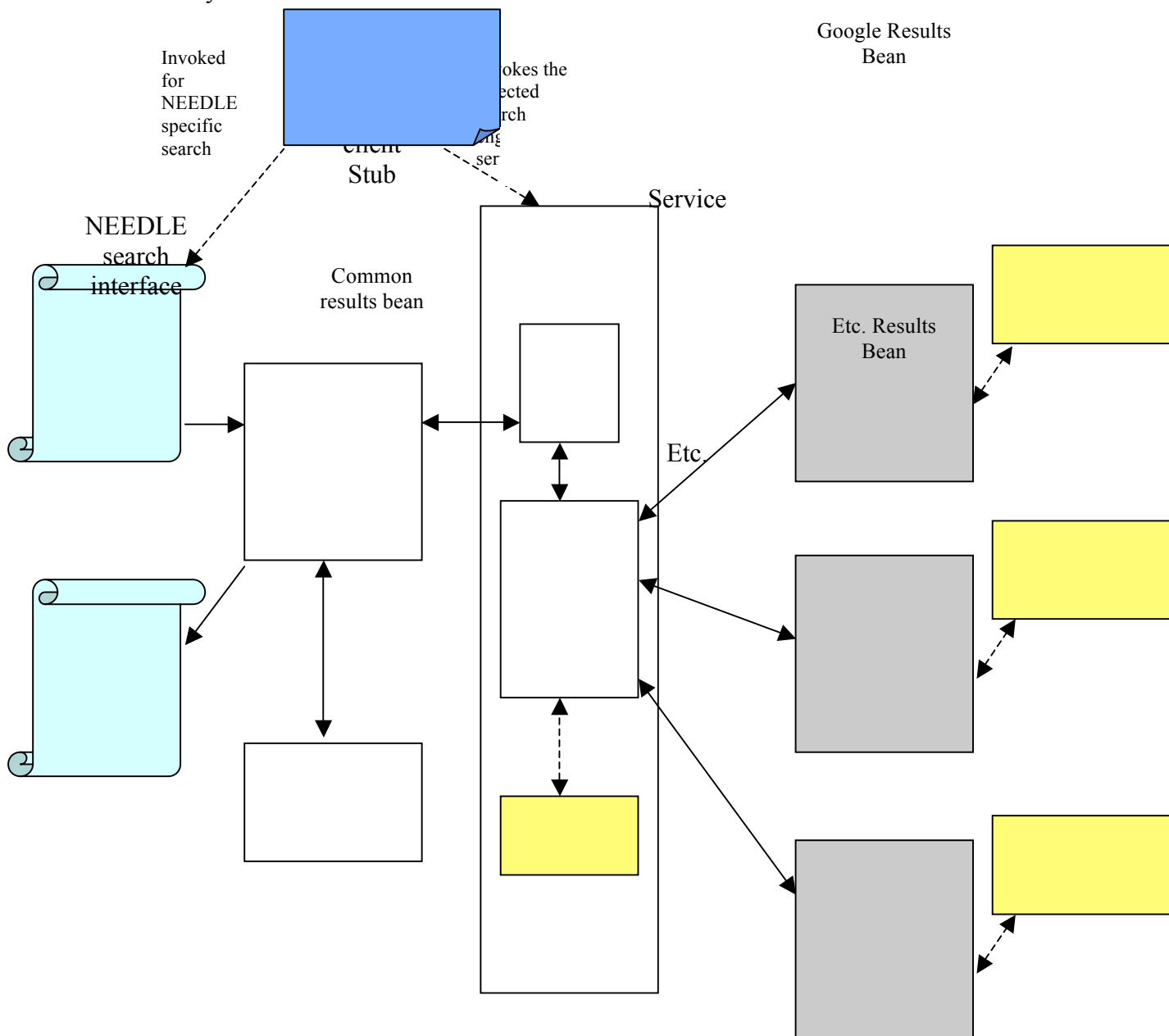


Fig. 7.7 Search Library with NEEDLE

7.6.4 Presentation Tier

As usual we use JSP page for getting the user input query and for displaying the results. An additional feature in the search screen is the option to select the desired Search Libraries. These are displayed as image or icons and the users can drag them and place into a search selection box. These image or icon details are fetched from the *Search Library Configuration File* which is an XML file (more details in the next section). Therefore, only the icons of configured Search Libraries appear in the screen.

Ajax technology is used to provide a more dynamic and user interactive search interface. For e.g. the drag and drop actions of the icons are achieved using Scriptaculous Ajax library. Details about Ajax technology has been discussed in Chapter 6.

The Servlet receives the user data from the JSP screen and pre-processes it to determine the selected Search Libraries. If, NEEDLE is selected then the query is sent to the standard NEEDLE search business logic and the corresponding flow is carried over. For other libraries, the Servlet invokes the client for invoking the corresponding web service.

7.6.5 Search Library Configuration File – XML File

Before moving on to the next step in the control flow, we discuss about the Configuration file which plays an important role in the business logic flow. We mentioned earlier that the users should be able to configure new Search Libraries. These configuration settings are stored in a special XML file called the *Search Library Configuration File*. As we now understand that every Search Library would be a new web service that interacts with NEEDLE through SOA, we need to store the details about the web service and the methods to invoke on these services. All these details are stored in the XML file. Every time a search is performed on a particular Search Library, the details are fetched by the search client from the XML file to invoke the corresponding web service. This means that the data found in the XML file determines the business flow in a dynamic fashion.

The following details are found in the XML file for each Search Library.

- 1) *Name* – This is the name of the Search Library e.g. Google
- 2) *Image* – The filename of the image to be displayed
- 3) *Stub class name*– The stub which contains the code to invoke the remote web service of the search engine. More details about the stub generation are discussed in the Implementation section.

- 4) *Search method name* – The name of the particular search method that contains the business logic to accept the query and perform the search and return the results.
- 5) *Result bean class name* – This is the class name of the search result bean which contains the data members specific to the search engine.
- 6) *Application key* – Most search engines like Google, Yahoo etc. require an Application key to be registered and used every time we invoke the web service offered by them. The usage has some limitations on the number of hit results per query and number of searches per day etc.
- 7) *Result Field methods* – As of now, we have decided to display the following three fields as part of the search results
 - i. Title – Title of the web page
 - ii. Summary – Text summary (which consists of few sentences) of the particular search result.
 - iii. URL – URL of the web page

Some search engines may return more details for a search result. But, we have standardized the result to display only these specific fields. Therefore, we need to fetch these three data from the search engine specific format (search engine bean getter method) and set it into the standard format (common bean setter method). Hence, we store the names of the methods that give the values of these fields.

```
<Library type="web service">
  <Name>Google</Name>
  <Image>google.gif</Image>
  <Stub>samples.quickstart.GoogleSearchServiceStub</Stub>
  <StubSearchMethod>samples.quickstart.GoogleSearchServiceStub$searchGoogle</StubSearchMethod>
  <StubResultBean>samples.quickstart.GoogleSearchServiceStub$GoogleResultBean</StubResultBean>
  <ApplicationKey>Knzsc1pQFHKsbCu2ScZc+QEGcETZNhIb</ApplicationKey>
  <TitleMethod>getTitle</TitleMethod>
  <URLMethod>getLink</URLMethod>
  <SummaryMethod>getSnippet</SummaryMethod>
</Library>
```

Fig. 7.8 Snippet of Search Library XML Configuration File

7.6.6 Search Client

The Servlet invokes a client which does the following tasks in the same order,

- 1) Receives the list of Search libraries selected by the user.
- 2) For each Search Library, it fetches the web service details (stub class, search method name, result bean class etc.) from the Configuration file.
- 3) Invokes the corresponding stub class, which in turn invokes the web service of the search engine. This is done in a dynamic fashion through the use of *Java Reflections API*.
- 4) The results from the web service come in the form of search engine specific bean objects. Hence, the client fetches the results and sets it into the common result bean.
- 5) Finally sends the result bean to the Servlet to be processed and displayed to the user.

7.6.7 Web Service

The web service is just a normal java class developed using Axis2 API and deployed as Axis Archive (.aar) file. The web service class contains the search method which finally invokes the search engine web service API using the Application key. The search results are parsed and set into the corresponding bean objects and finally returned to the client.

The following are the reasons for making this component as a web service instead of a simple java class,

- 1) Developers could independently develop a web service with a search method and a result bean class and deploy it in Axis2 SOAP engine. More details on this are discussed in the implementation part.
- 2) In future, if a search library is added or modified or deleted, we need not touch the core J2EE components of NEEDLE system like JSP, Servlet etc. We just need to change the individual web service and the corresponding configuration settings in the XML file.
- 3) In future, even if NEEDLE system migrates from J2EE based platform to some other platform, the web services could still be invoked in the same manner as before, because SOA supports cross platform compatibility.

7.6.8 Implementation

Now we discuss the step by step procedure to develop a Search Library Web Service and integrate it with NEEDLE.

1) Development of Web Service

The web services used in this application are developed using Axis2 API and deployed in the Axis2 SOAP engine running on Jakarta Tomcat web container. Therefore, before developing the web service it is important to include the Axis2 library into the project. More details and steps on web service development using Axis2 was discussed in the section 7.1. The below steps are to be followed to develop the search engine web service,

- ✓ The first step is to write a simple Java class with a recommended standard name '*<Search Engine Name>WebSearch*', e.g. 'GoogleWebSearch'.
- ✓ The class contains the search method with a recommended name '*search<Search Engine Name>*', e.g. 'searchGoogle'. It accepts a query string and application key input arguments. This method contains the logic for invoking the search engine specific API.
- ✓ Create a bean class with data members as the search engine result fields, like title, summary, URL etc. The data members have corresponding getter and setter methods. The recommended name for the bean class is '*< Search Engine Name >ResultBean*', e.g. 'GoogleResultBean'. The recommended names for the getter and setter methods of data fields are '*getTitle()*', '*getSummary()*', '*getURL()*'.
- ✓ Create a '*services.xml*' file with details like Service name, Service class name (e.g. GoogleWebSearch).
- ✓ Create a '*build.xml*' to compile the class files and produce an Axis2 Archive file (.aar).

Thus, we have created a standalone web service archive file which is ready to be deployed. For further details regarding Axis2, please refer to section 7.1.

2) Deployment of Web Service

The .aar file created in the previous step has to be deployed in the Axis2 SOAP engine running on Tomcat server. To do this, simply copy the .aar file and place it in the following directory structure.

Tomcat root

└ webapps

└ axis2

└ WEB-INF

└ services

└ *Place it here (e.g.*

GoogleSearchService.aar)

Once the service has been placed into the above mentioned folder, start the Tomcat server and access the Axis administration console. If the service was successfully deployed, it would be listed in the console with other services. It is also possible to view the WSDL file of the deployed service.

3) Generation of Stub class

Now that we have developed a web service and successfully deployed it, we can invoke it for performing the search operations. However, in order to invoke the web service, we need a stub class which is responsible for packing and unpacking the data sent and received from the web service. It uses an Axiom based approach for exchanging data. The data exchange takes place through SOAP messages (section 7.1).

The stub class could be generated from the WSDL file that we have for the deployed service. The WSDL to Java code generation is done using the Axis2 Codegen plug-in for Eclipse. The detailed steps are given in section 7.1.

4) Integration with NEEDLE

After all these steps, we now have the completed Search Library web service, the corresponding stub class and other related generated classes. The final task now is to integrate the Search Library with NEEDLE. This could be done by updating the Search Library Configuration XML File with the details of the new web service. However, it is very difficult for a developer to manually add a new element to the XML file with all the details about the web service class, bean name etc. Therefore, we need a tool to automate this task of integrating. We call this a '*Search Library Customization tool*'.

7.7 Search Library Customization Tool

7.7.1 Motivation

The main objective of the tool is to help the developer to easily integrate the Search Library web service and the related stub class with NEEDLE. The following are the disadvantages of manually updating the Configuration XML file,

- 1) Difficult to manually enter XML tags elements with lengthy and confusing class names and stub file names.
- 2) The developer has to open the stub file and read through it to find the fully qualified class name of the result bean class and the method names.
- 3) Possibility of making errors while entering manually.

Therefore, it would be very useful to develop a tool which the developer can use as a ‘wizard’ for integration.

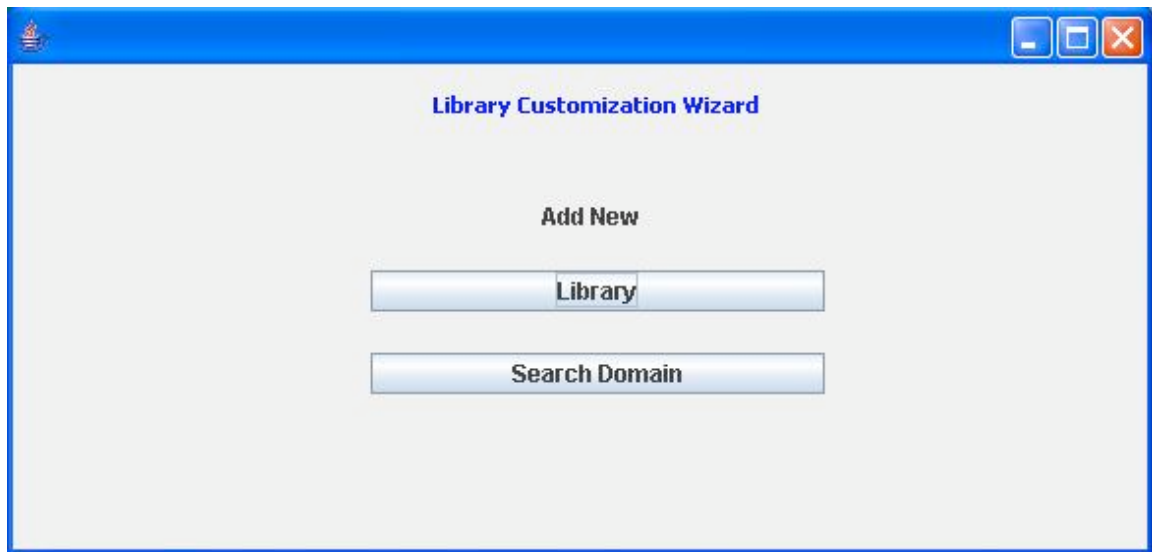
7.7.2 Requirements

The following are the functional requirements for the Customization tool,

- 1) The tool should be a standalone application and be executed from the development environment e.g. from Eclipse IDE. In short, the tool should be a wizard.
- 2) Should be able to configure two types of Search Libraries namely, Search Engine web service and Search domain.
- 3) Should collect all the details required by the Configuration file like Search library name, stub class, application key etc. The class names and method names should be provided as drop down selection to help the user avoid typing.
- 4) Finally, update the XML file with the given details.

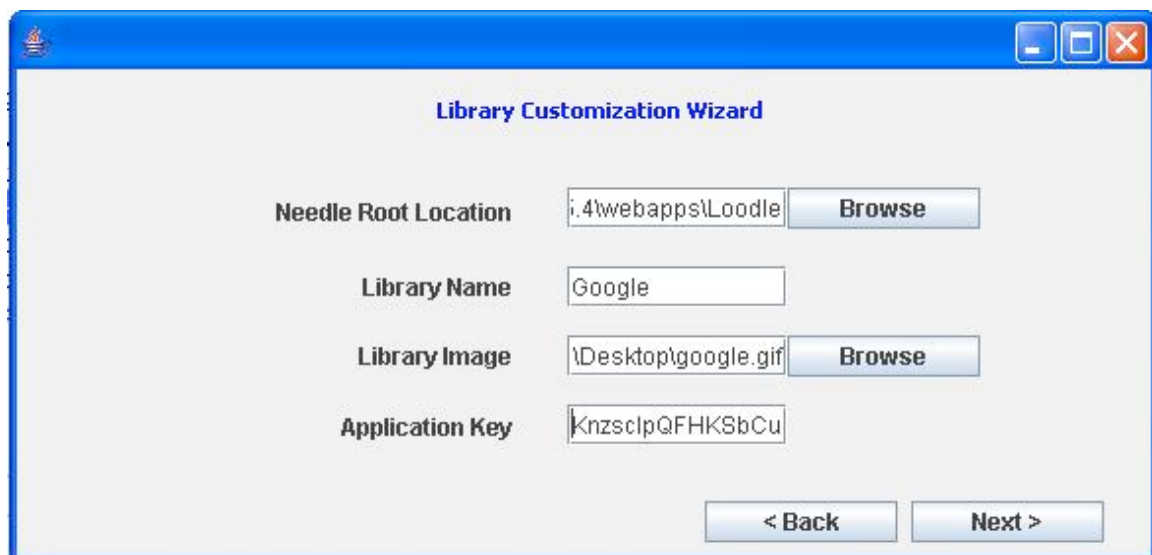
7.7.3 Detailed Screen Description

1) Search Library Type Selection Screen



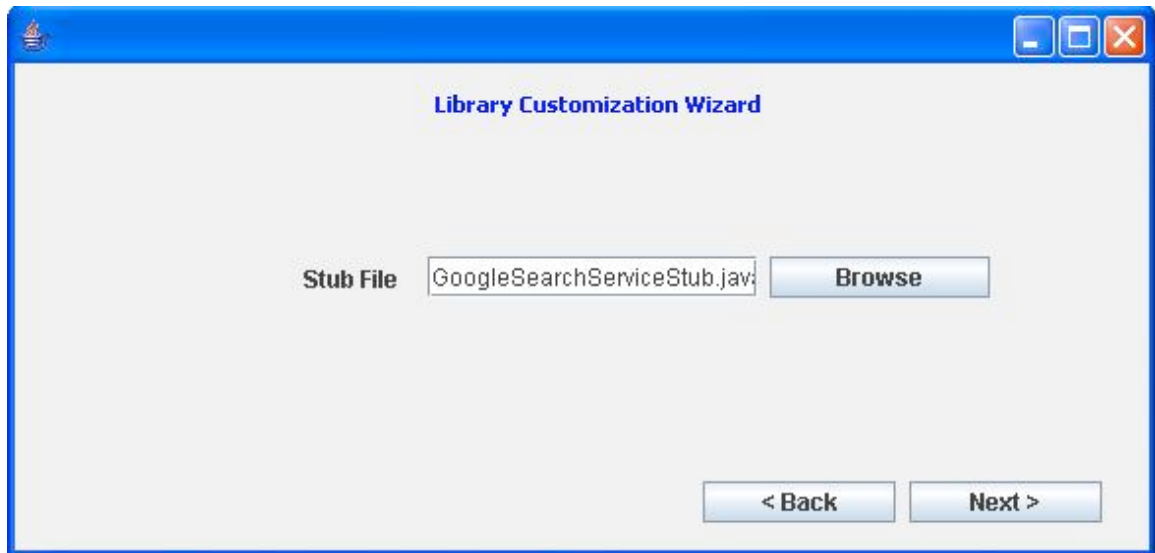
The user could either choose to configure / customize a Search Library or a Search Domain.

2) General Details screen



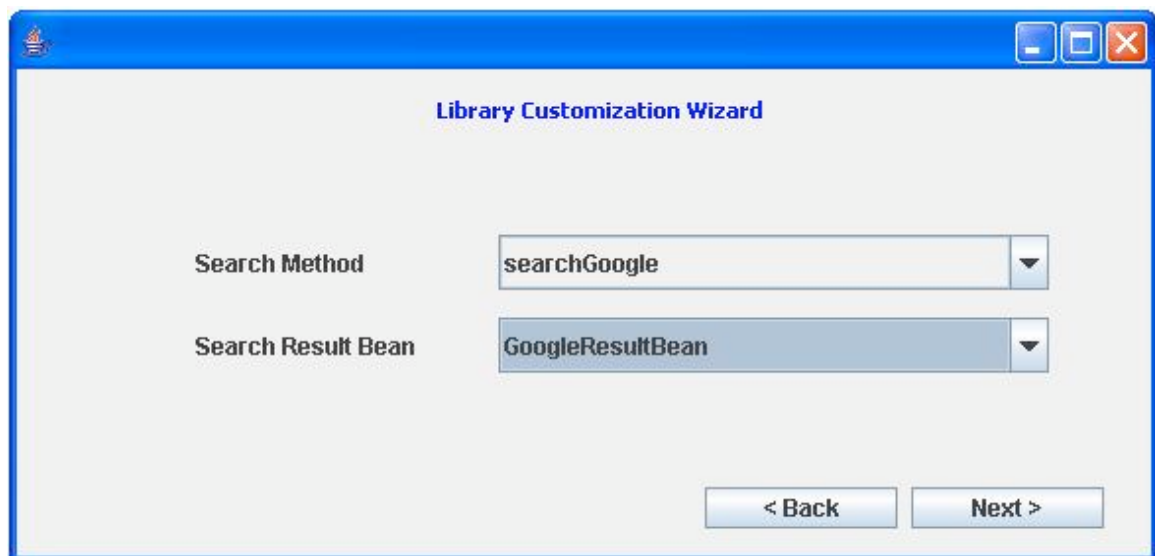
In this screen, the user enters the general details like NEEDLE root location in Tomcat webapps folder, Library name, image file location and Application key.

3) Stub class Selection Screen



The user could use the Browse button to select the stub file (.java)

4) Search Method and Result Bean Selection Screen



The tool automatically displays all the methods in the selected stub class. The user could select the relevant search method. Similarly, the user selects the result bean class name.

5) Getter Method selection for Result Bean fields



The screenshot shows a window titled "Library Customization Wizard" with a standard Windows-style title bar (minimize, maximize, close buttons). Inside the window, there is a table with two columns: "Field" and "Method". The table contains three rows of data. The first row shows "Title" mapped to "getTitle". The second row shows "URL" mapped to "getUrl". The third row shows "Summary or Snippet" mapped to "getSnippet". At the bottom right of the window, there are two buttons: "< Back" and "Finish".

Field	Method
Title	getTitle
URL	getUrl
Summary or Snippet	getSnippet

< Back Finish

The web service returns the search engine specific result fields. However, we limit the results to just three fields like title, URL and summary. In order to set the values into the common result bean, we need to map the corresponding getter methods present in the search engine result bean. This screen is used for that purpose.

Finally, after clicking the 'Finish' button, the tool processes the input and updates the Configuration file.

Chapter 8:

Future Work

8 Future Work

All through the work we have discussed various literature and experimented different technologies to implement the required functionalities. As a result of such study, we have identified some of the key areas to be developed in the future. We present them as potential future work for NEEDLE.

Development of Web Services

In this thesis work, we have successfully introduced the concept of “Search Library” in NEEDLE which is based on Service Oriented Architecture (SOA). As a result, NEEDLE has evolved into the new generation distributed computing architecture which is open to collaborate with external applications. However, this is an initial step only. NEEDLE could develop and expose more e-Learning specific web services that could be used by other external applications to use the huge collection of NEEDLE’s multimedia e-Learning materials.

The E-Learning Framework (ELF - <http://www.elframework.org>) [8] that we discussed in Chapter 3 proposes a standardized, open architecture for development and interaction between e-Learning applications. NEEDLE could be used as a rich source of video lectures and presentation slides that might be useful for external learners. In the opposite direction, NEEDLE could use the services exposed by other applications and thus extend the functionalities. As of now, we have implemented “Search Libraries” only for standard search engines. In future, more number of digital repositories could be added in the search domain.

Thus, by following a SOA based approach, NEEDLE achieves a mutual benefit.

Semantic Searching

The current search engine for NEEDLE works with raw text based search. It has proved to be simple and efficient. However, with the increasing volume of heterogeneous multimedia data, it becomes necessary to organize and classify them for efficient retrieval.

The recent research work on NEEDLE aims at extracting the concepts and semantic relationships from the multimedia e-Learning materials to develop Ontologies and classify them accordingly. Similar ideas were discussed in ATLAS [6] application in Chapter 3.

By using such semantics and Ontologies, intelligent searching could be performed. The application would be able to identify the real need of a user when he

enters a search query and accordingly suggest more options to discover the correct resources.

Personalization and User Profiling

Personalization is very important in e-Learning domain. The users should be able to configure the user interface, set preferences etc. according to their standard/level of learning. These preferences could be stored as User-Profiles and the functionalities like content delivery, searching and ranking etc. could be adapted based on the settings.

The configuration of “Search Libraries” could be considered as user personalization, because the user can configure the preferred search engines and are free to develop their need specific web service implementations. Also, only the configured search engines are displayed on the search screen and could be selected by the user for searching.

The concept of personalized search could be extended to provide ranking and customized display of search results. For e.g. the user could assign scores or weight age to certain subjects that he / she prefers to be ranked more than others. Accordingly, the ranking and display of results could be sorted to help the user to find the best hits on top of others.

The above mentioned areas are currently under research and very soon the prototype would contain such advanced features. Many other similar ideas are under consideration and need to be explored to enhance the NEEDLE application.

Chapter 9:

Conclusion

9 Conclusion

In this thesis work we have worked with one of the modern, evolving, web-based e-Learning application called NEEDLE which provides a search interface for discovering resources from a rich collection of multimedia e-Learning materials. The NEEDLE application depends on LODE for receiving video lectures and synchronized PowerPoint slides.

After careful system study and related literature review, we worked on re-engineering the prototype to add some functionalities and enhance few others. Specifically, we worked on the re-design of the prototype to follow better J2EE design patterns, changed the data access layer and implemented Hibernate technology for its advantages over JDBC. We also introduced advanced search functionalities to help the user to refine the query and narrow the search results.

In parallel, we also developed an automation tool called “NEEDLE Upload Tool”, for processing the data from the LODE system and prepare it to be searchable from NEEDLE. The tool could be used as a standalone application to upload any new lecture series to the NEEDLE backend. The procedure which was previously done manually was error prone and time consuming. The tool automates the complete process and proves very helpful for the users.

NEEDLE has now moved into the next generation of distributed applications by the usage of Service Oriented Architecture (SOA). The concept of “Search Library” allows the user to search for materials outside NEEDLE repository. A Customization tool was developed to assist the developers to integrate the newly created search web services with the NEEDLE application. This tool takes care of mapping the web service specific classes like stub class etc. into a XML configuration file which is used by NEEDLE to invoke the respective Search Library.

All the prototypes and tools developed during this thesis work were tested and evaluated by receiving feedback from users and developers and incrementally enhanced.

It was a great learning experience to work with different design architectures and to experiment the latest state-of-the-art-technologies. The thesis gave a very good exposure to the development of a complete web-based application based on Java. As a conclusion, the thesis provided a right mix of theoretical study and practical hands on development experience.

APPENDIX

APPENDIX - A

NEEDLE Upload Tool

Installation & Usage

The Needle Upload Tool is packaged as archive and contains the source, compiled class files, required libraries and resources for uploading the e-learning contents. The following steps are to be carried out,

1. Unzip the contents of the *NeedleUploadTool.zip* to a directory
2. Edit the *run.bat* inside the NeedleUploadTool directory
 - 2.1 Edit the NEEDLE_LIB variable to the location of the NeedleUploadTool.
e.g. D:\temp\NeedleUploadTool\
3. Open *LOODLE_PROP.properties* and edit DATA_ROOT and LUCENE_ROOT variables to point to the desired locations.
4. Execute *run.bat*

content

data

content_frame

collection of images

Input Data Archive

title.html

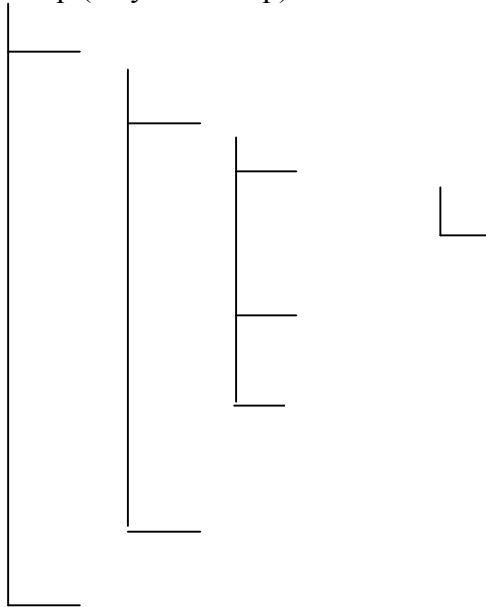
The user is supposed to provide as input the video lectures and other meta-data that is produced by the LODE system. These files should be packaged as a .zip file and the directory structure present inside this archive (.zip file) is very important. It is assumed that the input file obeys this structure and the process is executed. Any difference in the structure from the one discussed here, would result in failure of the process. The names in bold and italic should be present as given.

movie.mp4

index.html

common

XYZ.zip (Any name .zip)



BIBLIOGRAPHY

Bibliography

- [1] Giuseppe Riccardi, Marco Ronchetti, "*NEEDLE: Next gEneration sEarch engine for Digital LibrariEs*", In Proceedings of AISV, TRENTO, 2006.
- [2] Dolzani M., Ronchetti M., "*Video Streaming over the Internet to support learning: the LODE system*". WIT Transactions on Informatics and Communication Technologies, 2005, v. 34, p. 61-65
- [3] Dolzani, M., & Ronchetti, M. (2005). "*Lectures On DEmand: the architecture of a system for delivering traditional lectures over the Web*". In Kommers, P., & Richards, G. (Eds.), Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2005 (pp. 1702-1709). Chesapeake, VA: AACE.
- [4] Ronchetti M., "*Has the time come for using video-based lectures over the Internet? A Test-case report*", CATE - Web Based Education Conference 2003, Rhodes (Greece), June 30 - July 2, 2003
- [5] Ronchetti, M. "*Using the Web for Diffusing Multimedia Lectures: A Case Study*". In Kommers, P., & Richards, G. (Eds.), Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2003 (pp. 337-340). Chesapeake, VA: AACE.
- [6] Marc Spaniol et al., "*ATLAS: A Web-Based Software Architecture for Multimedia E-learning Environments in Virtual Communities*", International Conference on Web-based Learning (ICWL 2003), Springer-Verlag, LNCS 2783, pp. 193-205, 2003.
- [7] Victor Manuel Garcia-Barrios et. al, "*AdELE - Adaptive E-Learning through Eye Tracking: Theoretical Background, System Architecture and Application Scenarios*", Proceedings of International Conference on Knowledge Management (I-KNOW'04), Graz, Austria, 2004.
- [8] "*E-Learning Framework*" - <http://www.elframework.org>
- [9] B. Low, J. MacColl, "*Searching Heterogeneous e-Learning Resources*", paper presented at the DELOS Workshop, Digital Repositories: Interoperability and Common Services, May, 2005.
<http://delos-wp5.ukoln.ac.uk/dissemination/pdfs/low.pdf>
- [10] Martin Bond et. al., "*Teach Yourself J2EE in 21 Days*", First Edition, SAMS Publishing, 2002.
- [11] Eric Armstrong et. al., "*The J2EE 1.4 Tutorial*", Sun Microsystems, 2003.

- [12] D.K. Fields & M.A. Kolb, “*Web Development with Java Server Pages*”, First edition, Manning Publications, 2000.
- [13] “*Servlet Life Cycle*” – www.java.sun.com/javaee/5/docs/tutorial/doc/Servlets4.html
- [14] Ed Roman et. al., “*Mastering Enterprise JavaBeans*”, Second edition, Wiley Computer Publishing, 2002.
- [15] Inderjeet Singh et. Al., “*Designing Enterprise Applications with the J2EE Platform*”, Second Edition, Addison-Wesley, 2002.
- [16] “*Hibernate Vs JDBC*” - www.mindfiresolutions.com/download/Java=Hibernate_JDBC.pdf
- [17] “*Apache Lucene*” – <http://lucene.apache.org>
- [18] “*Lucene*” - <http://www.javaranch.com/newsletter/200404/Lucene.html>
- [19] “*QuickTime Overview*” – <http://developer.apple.com/documentation/QuickTime/RM/Fundamentals/QTOverview/QTOverview.pdf>
- [20] “*MP4BOX Documentation*” – http://gpac.sourceforge.net/doc_mp4box.php
- [21] Bernd Bruegge & Allen Dutoit, “*Object-Oriented Software Engineering: Using UML, Patterns, and Java*”, Prentice Hall, 2003.
- [22] “*Learn the NET: Advanced Web Searching Techniques*” - <http://www.learnthenet.com/english/html/77advanc.htm>
- [23] “*Advanced Searching Techniques*” - <http://library.duke.edu/services/instruction/libraryguide/advsearch.html>
- [24] “*Apache Lucene*” – <http://lucene.apache.org>
- [25] N. C. Zakas, J. McPeak, J. Fawcett, “*Professional Ajax*“, Wrox Press, London, 2006.
- [26] “*Ajax Tutorial*” - www.w3schools.com/ajax/default.asp
- [27] “*Asynchronous JavaScript Technology and XML (Ajax) With the Java*” – www.java.sun.com/developer/technicalArticles/J2EE/AJAX/
- [28] “*Web Services Architecture Overview*” - www.ibm.com/developerworks/library/w-ovr/

- [29] “*Web Services Architecture*” - www.w3.org/TR/2003/WD-ws-arch-20030514/
- [30] “*Axis 2.0 - Axis2 User's Guide*” –
www.ws.apache.org/axis2/0_94/userguide1.html
- [31] “*Axis2/Java - Code Generator Wizard Guide for Eclipse Plug-in*” -
www.ws.apache.org/axis2/tools/1_2/eclipse/wsd2java-plugin.html
- [32] Richard Monson-Haefel, “*J2EE Web Services*”, First edition, Addison-Wesley, October 17, 2003
- [33] Thomas Erl, “*Service-Oriented Architecture: Concepts, Technology, and Design*” Prentice Hall, 2004.
- [34] “*The Java Web Services Tutorial*” –
www.java.sun.com/webservices/tutorial.html

Updated Document

NEEDLE Index Structure

With an objective to include ranking of search hit results, there has been a change in the NEEDLE indexing structure and search procedure.

Ranking

Lucene indexing API provides a mechanism to include TF-IDF (Term frequency- Inverse Document Frequency) measure for the words that are indexed. And also additional boosted values i.e. weight age is given to important words and to words found on titles etc. These values help to determine the importance of a particular lecture.

Hence, the search hits are ranked according to the lecture importance based on TF-IDF measure.

Indexing

In order to achieve the above mentioned ranking, we have used two separate indexes,

- General Index
- Lecture Specific Index

General Index

This is used to index the text of both audio transcript and slide with one Lucene '*Document*' for each lecture. The index structure is as follows,

Series ID	ID of the lecture series or course
Event ID	ID of the particular lecture
Text	Text of both audio transcript and slide content
Titles	Titles of presentation slide. Has more boost for the TF-IDF value to stress importance.

The TF-IDF measure is calculated for words in this index.

Lecture Specific Index

This is used to store the text of both audio transcript and slide with one Lucene *'Document'* for each timestamp (for audio transcript) and Lucene *'Document'* for each slide (for slide text). The index structure is as follows,

Series ID	ID of the lecture series or course
Event ID	ID of the particular lecture
Markup	Timestamp of the video (null for Slide data)
Number	Slide number (null for video data)
Media	Either “video” or “slide”
Transcript Text	Text of audio transcript
Slide Text	Text of slide content
Slide Title	Text of slide title

This index is stored inside specific directory structure corresponding to the series ID and event ID of the lecture. The directory structure is as follows,

LUCENE_LECTURE_INDEX_ROOT / seriesID / eventID / Lucene /

Search Procedure

The following is the new search procedure,

1. The query text is first searched in the “General Index”. The hits are ranked according to the importance of lectures.
2. The hits are iterated and the Series ID and Event ID are fetched for the hits to form directory path strings corresponding to the lecture specific index. These path strings are returned as a List.
3. The List of path strings is iterated and the particular lecture index is searched.
4. The hits contain details about the video timestamp, slide number etc.
5. The hit details are collected and stored in data transfer objects and sent back to web-tier.

NEEDLE Upload Tool

The tool has been updated to create two indexes as described above.

Updates to Properties File

The following two entries are to be added to the LOODLE.properties

LUCENE_GENERAL_INDEX_ROOT // Points to general index location.

LUCENE_LECTURE_INDEX_ROOT // Points to lecture index location.