|  |
| --- |
| **TEAM - iNCREDIBLE Course - CMPE 276 (Spring 2012)**  **XML Technologies**  sjsu |
| Online Book Catalogue |
| Final Report |
|  |
| Anuj Acharya, Sagar Zagde Duy Nguyen, Anurag Gade, Tom Tran, Sai Krishna |
| **4/6/2012** |

|  |
| --- |
|  |

**Table of Content**

[1 Introduction 3](#_Toc321489581)

[2 System Architecture 4](#_Toc321489582)

[3 eXist-DB XML Database 5](#_Toc321489583)

[3.1 Final Schema BookCatalogue.xsd 5](#_Toc321489584)

[3.2 XMLSpy Graphical View 7](#_Toc321489585)

[3.3 Sample XML Documents 8](#_Toc321489586)

[3.4 Google Book API 9](#_Toc321489587)

[3.5 JAX-B API 9](#_Toc321489588)

[3.6 eXist-DB Collection 11](#_Toc321489589)

[3.7 eXist-DB Rest API 12](#_Toc321489590)

[4 Solr-Lucene Search Engine 13](#_Toc321489591)

[4.1 DataImportHandler 13](#_Toc321489592)

[4.2 solrconfig.xml 13](#_Toc321489593)

[4.3 exist-config.xml 13](#_Toc321489594)

[4.4 schema.xml 14](#_Toc321489595)

[4.5 Scheduling Import with DataImportScheduler 15](#_Toc321489596)

[5 Features of On-Line Book Catalogue 15](#_Toc321489597)

[5.1 Home Page Navigation 15](#_Toc321489598)

[5.2 Browse Books 16](#_Toc321489599)

[5.3 Aside Links 17](#_Toc321489600)

[5.4 Navigation of Pages and Summary related feature 19](#_Toc321489601)

[6 HTML5 and CSS3 Features used 21](#_Toc321489602)

[7 Problem & Enhancement 23](#_Toc321489603)

[7.1 Problem with XPathEntityProcessor 23](#_Toc321489604)

[7.2 XML Importer does not have incremental import 23](#_Toc321489605)

[8 Conclusion 24](#_Toc321489606)

[9 References 25](#_Toc321489607)

# Introduction

Following the design from the previous lab, we continued to expand on the same online book catalog concept and implemented the entire web application. Both the database and the user interface were developed in this project but with some minor changes compare to what was previously planned. Instead of having the user login, user can now directly enter the site through any browser and search for books based on name, author, and ISBN. Upon entering the site, user will be presented with top author and top publisher listing for them to choose from. Following the requirements, the database was developed using XML format and were stored in eXist. Solr was used for indexing all the XML files in the database for quick search. Finally, we used HTML5 along with CSS to develop the user interface that could be opened and aligned nicely with Google Chrome. Due to limited time, we were unable to implement some minor features such as shopping cart, finding local store, and online audio book.

# System Architecture

Solr used to build indexes

eXist

(Manage XML documents)

HTML5 application

DESCRIPTION:

eXist is a tool used to store the XML documents acting like database. The Solr tool is used to index and store the XML documents for faster access of the data. Ajax-solr is used to make the connection between the Solr and the HTML5 application. The data is populated in HTML file using ajax-solr APIs.

# eXist-DB XML Database

Instead of using the traditional relational database such as MySQL or SQL server to store the book information, this web application used XML documents to store those books information. Each XML document conforms to BookCatalogue.xsd schema and each document represents a book. Finally, these XML documents are stored and managed by eXist.

## Final Schema BookCatalogue.xsd

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

<!-- edited with XMLSpy v2010 rel. 3 (http://www.altova.com) by Duy Nguyen (Duy Nguyen) -->

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:bc="http://www.incredible.com/2012/BookCatalogue" targetNamespace="http://www.incredible.com/2012/BookCatalogue" elementFormDefault="qualified">

<xs:element name="Review">

<xs:complexType>

<xs:sequence>

<xs:element name="Reviewer" type="xs:string"/>

<xs:element name="ReviewText" type="xs:string"/>

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="Multimedia">

<xs:complexType>

<xs:sequence>

<xs:element name="IconURL" type="xs:anyURI" minOccurs="0"/>

<xs:element name="ImageURL" type="xs:anyURI" minOccurs="0"/>

<xs:element name="AudioURL" type="xs:anyURI" minOccurs="0"/>

<xs:element name="VideoURL" type="xs:anyURI" minOccurs="0"/>

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="BookReview">

<xs:complexType>

<xs:sequence>

<xs:element ref="bc:Review" maxOccurs="unbounded"/>

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="BookDetail">

<xs:complexType>

<xs:sequence>

<xs:element name="FullTitle" type="xs:string" minOccurs="0"/>

<xs:element name="PageCount" type="xs:int" minOccurs="0"/>

<xs:element name="Summary" type="xs:string"/>

<xs:element name="Subject" type="xs:string" minOccurs="1" maxOccurs="unbounded"/>

<xs:element name="Keywords" type="xs:string"/>

<xs:element name="BookFormat" type="xs:string" minOccurs="0"/>

<xs:element ref="bc:Multimedia" minOccurs="0"/>

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="BookCatalogue">

<xs:complexType>

<xs:sequence>

<xs:element ref="bc:Book" maxOccurs="unbounded"/>

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="Book">

<xs:complexType>

<xs:sequence>

<xs:element ref="bc:BookDetail"/>

<xs:element name="Author" type="xs:string" minOccurs="1" maxOccurs="unbounded"/>

<xs:element name="Publisher"/>

<xs:element ref="bc:BookReview" minOccurs="0"/>

</xs:sequence>

<xs:attribute name="Title" type="xs:string" use="required"/>

<xs:attribute name="ISBN" type="xs:string" use="required"/>

<xs:attribute name="Edition" type="xs:string" use="optional"/>

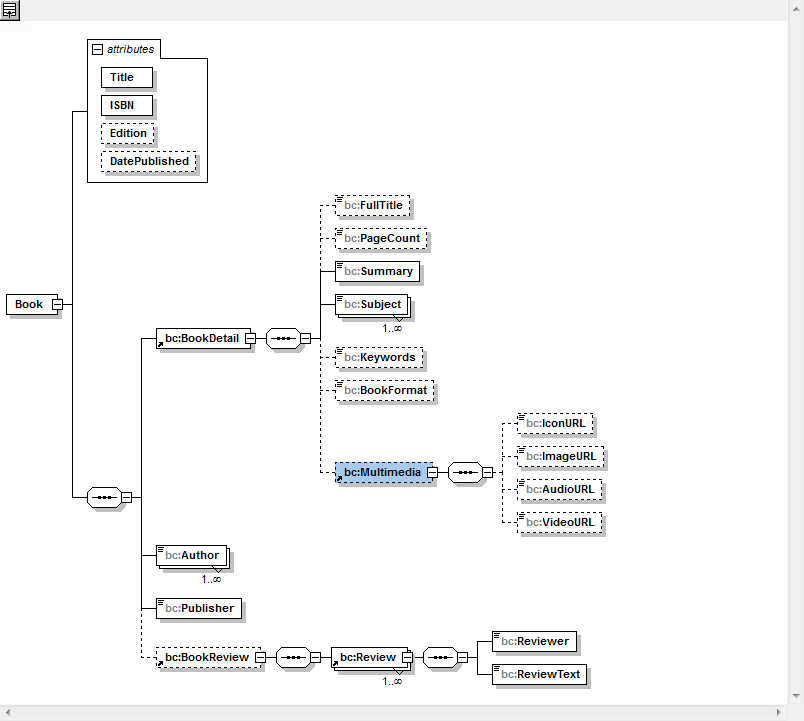
<xs:attribute name="DatePublished" type="xs:date" use="optional"/>

</xs:complexType>

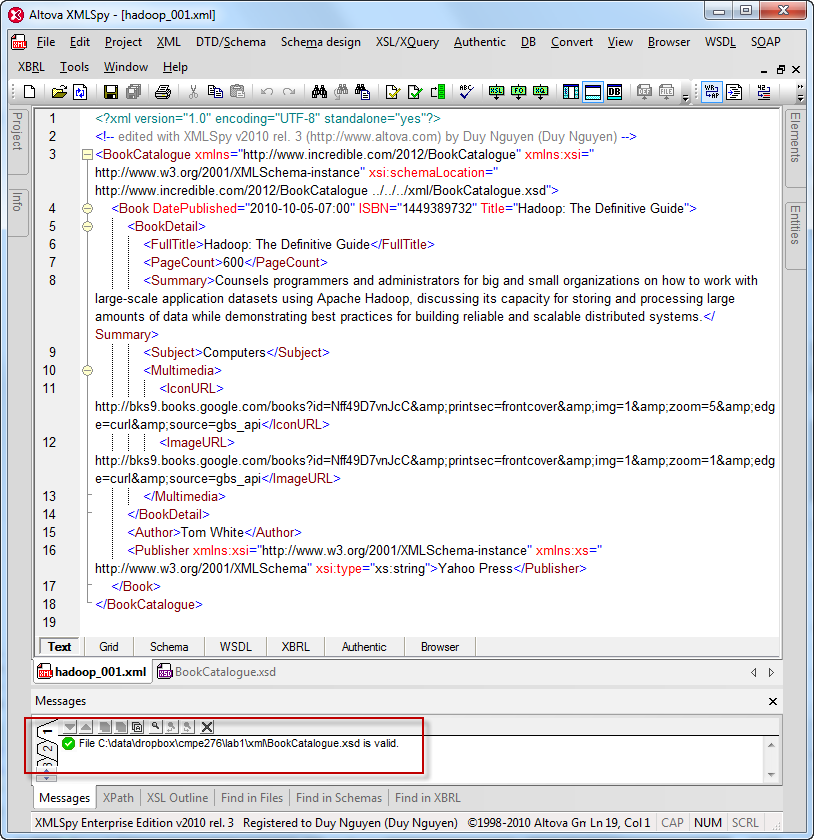
</xs:element>

</xs:schema>

## XMLSpy Graphical View

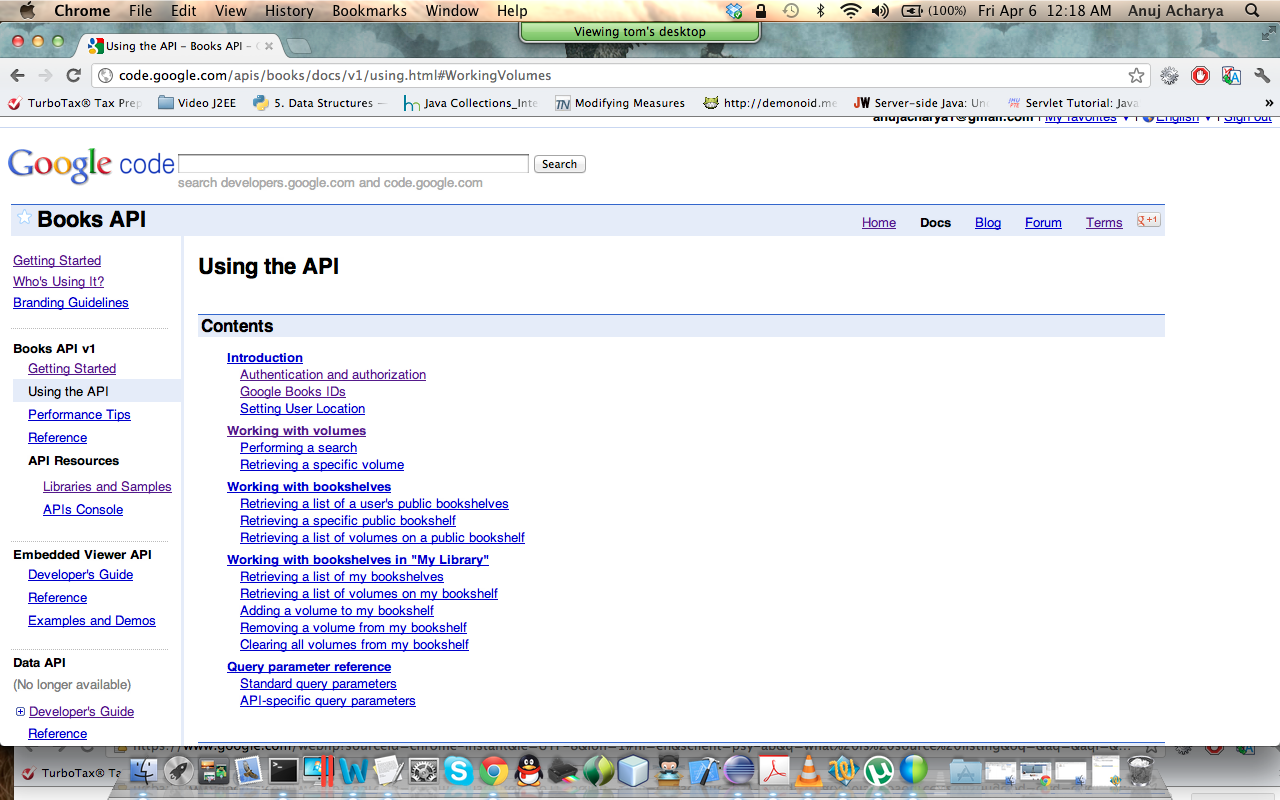


## Sample XML Documents

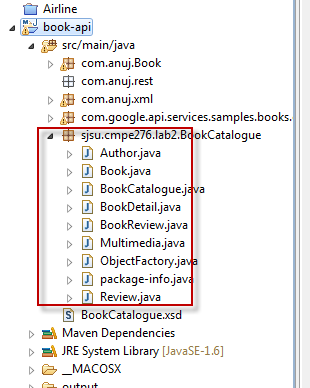


## Google Book API

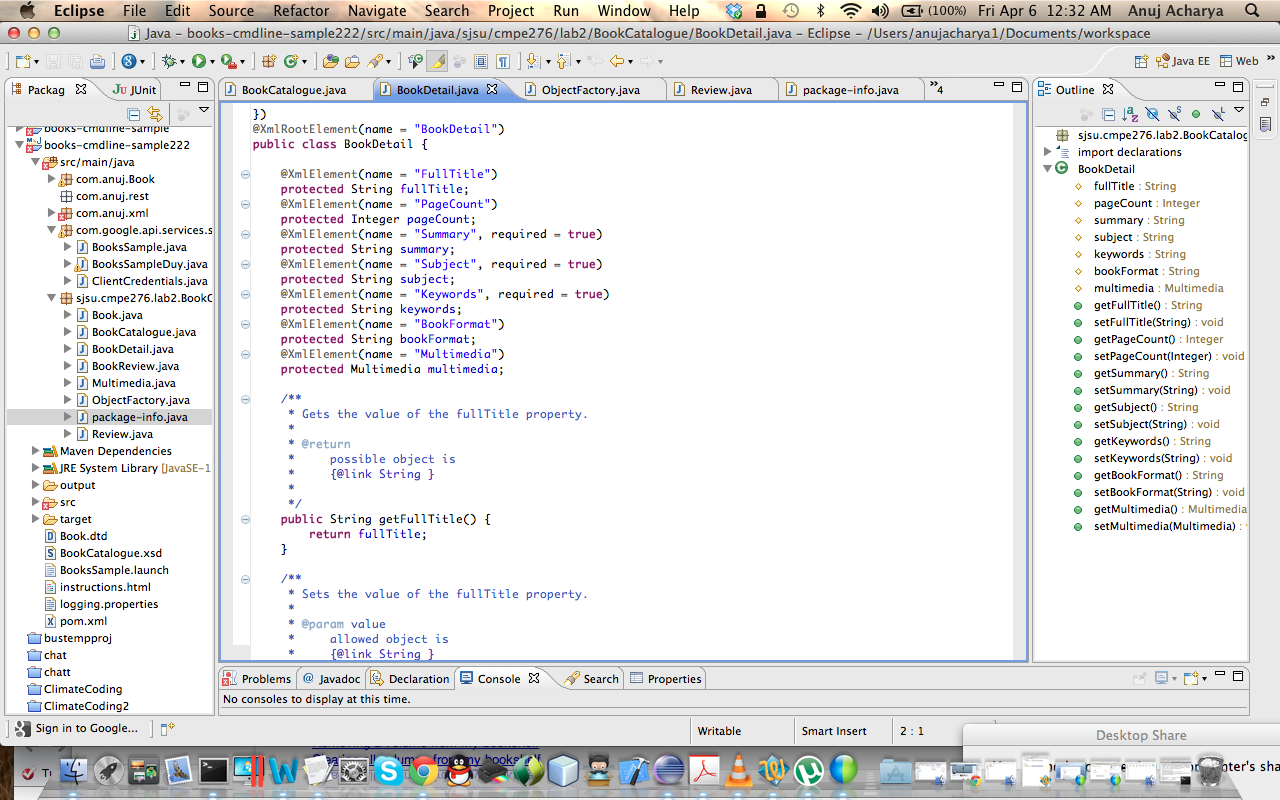
We used Google Books API to auto generate XML files based on the XML Schema we design. The data generated in the XML file is the up-to-date data of the books being sold on Google Books.



## JAX-B API

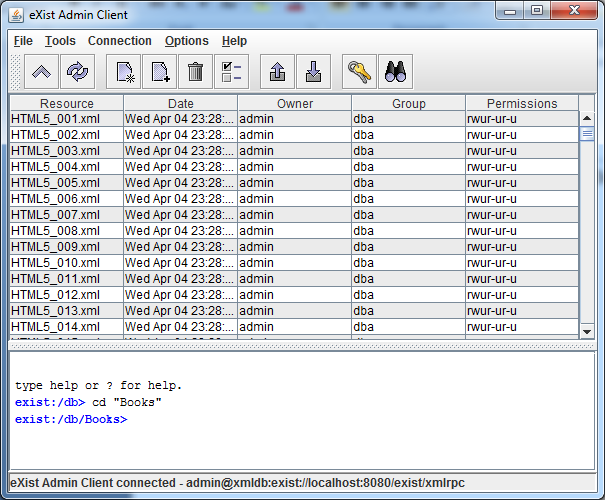
We took advantage of JAX-B API to generate a set of Java classes from our XSD file. With JAX-B we were able to map Google Book JSON object and generate over 1000 sample xml files in three program runs.

This XML files generated are uploaded to exist database that is index through Apace Solr for fast search query on the database.



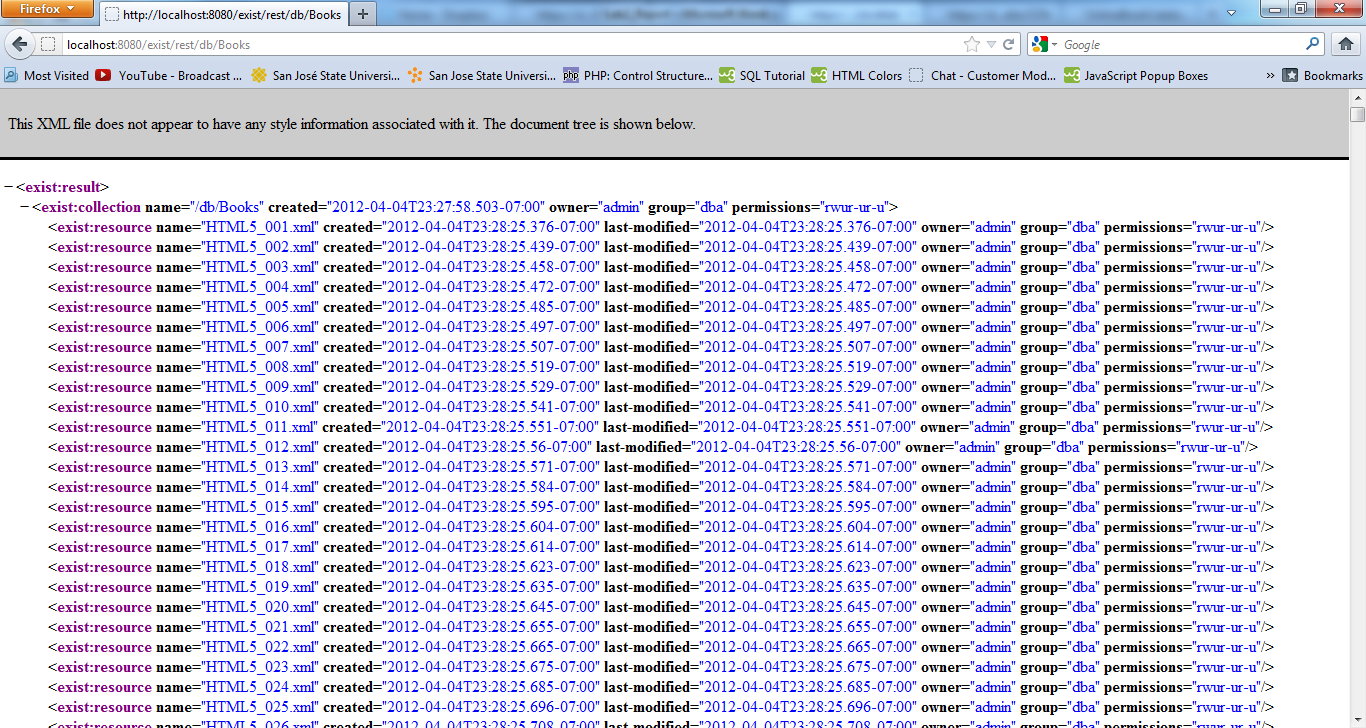
## eXist-DB Collection

We used eXist to store and manage all the book xml files. We loaded these xml files to eXist using eXist Admin Client provided as shown below. First we make a collection “Books” then we simply add all of these xml files to that collection.



## eXist-DB Rest API

After importing all the xml files to eXist, eXist will automatically hosts those xml file locally providing a restful service that can be accessed using HTTP requests at <http://localhost:8080/exist/rest/db>.



Since our database was stored in the Books collection, we will need to access that path in order to see all the book xml files.

# Solr-Lucene Search Engine

In order to provide user with complete control over the database, Apache Solr was used to index the book database and provides a complete full text search. User may search for a book based of title, author, or ISBN.

## DataImportHandler

This is Solr’s data import request handler that provides ways to import data from relational databases or XML files to Solr index. We used this request handler along with the full import command to import all the xml files hosted by eXist into Solr.

## solrconfig.xml

We added an additional request handler to this file known as *existimport* that uses the data import handler to import all the xml documents hosted by eXist at http://localhost:8080/exist/rest/db/Books into Solr index through URL based.

<requestHandler name="/existimport" class="org.apache.solr.handler.dataimport.DataImportHandler">

<lst name="defaults">

<str name="config">exist-config.xml</str>

</lst>

</requestHandler>

## exist-config.xml

This file was used to help Solr choose the correct path of where the XML database file is located and how to extract each document. Using Solr’s full import command <http://localhost:8983/solr/db/existimport?command=full-import>, we imported all the book XML documents from eXist to Solr index for search [1].

<dataConfig>

<dataSource name="eXist" type="URLDataSource" baseUrl="http://localhost:8080/exist/rest/db/" encoding="UTF-8"/>

<document>

<entity dataSource="eXist" processor="XPathEntityProcessor" name="BooksCollection" url="Books" forEach="/result/collection/resource" transformer="DateFormatTransformer">

<field column="xmlfile" xpath="/result/collection/resource/@name"/>

<entity dataSource="eXist" processor="XPathEntityProcessor" name="xmlfile" url="Books/${BooksCollection.xmlfile}" forEach="/BookCatalogue/Book" transformer="DateFormatTransformer">

<field column="title" xpath="/BookCatalogue/Book/@Title"/>

<field column="id" xpath="/BookCatalogue/Book/@ISBN"/>

<field column="isbn" xpath="/BookCatalogue/Book/@ISBN"/>

<field column="edition" xpath="/BookCatalogue/Book/@Edition"/>

<field column="date\_published" xpath="/BookCatalogue/Book/@DatePublished"/>

<field column="author" xpath="/BookCatalogue/Book/Author"/>

<field column="full\_title" xpath="/BookCatalogue/Book/BookDetail/FullTitle"/>

<field column="page\_count" xpath="/BookCatalogue/Book/BookDetail/PageCount"/>

<field column="summary" xpath="/BookCatalogue/Book/BookDetail/Summary"/>

<field column="book\_format" xpath="/BookCatalogue/Book/BookDetail/BookFormat"/>

<field column="subject" xpath="/BookCatalogue/Book/BookDetail/Subject"/>

<field column="keywords" xpath="/BookCatalogue/Book/BookDetail/Keywords"/>

<field column="publisher" xpath="/BookCatalogue/Book/Publisher"/>

<field column="reviewer" xpath="/BookCatalogue/Book/BookReview/Review/Reviewer"/>

<field column="review\_text" xpath="/BookCatalogue/Book/BookReview/Review/ReviewText"/>

<field column="icon\_url" xpath="/BookCatalogue/Book/BookDetail/Multimedia/IconURL"/>

<field column="image\_url" xpath="/BookCatalogue/Book/BookDetail/Multimedia/ImageURL"/>

<field column="audio\_url" xpath="/BookCatalogue/Book/BookDetail/Multimedia/AudioURL"/>

<field column="video\_url" xpath="/BookCatalogue/Book/BookDetail/Multimedia/VideoURL"/>

</entity>

</entity>

</document>

</dataConfig>

## schema.xml

We added additional coding lines to the original schema.xml file provided by Solr to force Solr to store these appropriate fields from our XML documents as shown below. We only index some important fields such as isbn or author because these fields are used for book searching whereas fields such as icon\_url or page\_count were not indexed but they will be stored in the Solr document.

Fields like author are type “string” so that Solr can skip the tokenization step that can break up the author name (ie: first, last).

<field name="subject" type="string" indexed="true" stored="true"/>

<field name="isbn" type="text\_general" indexed="true" stored="true"/>

<field name="edition" type="string" indexed="false" stored="true"/>

<field name="date\_published" type="string" indexed="false" stored="true"/>

**<field name="author" type="string" indexed="true" stored="true"/>**

<field name="full\_title" type="text\_general" indexed="true" stored="true"/>

<field name="page\_count" type="text\_general" indexed="false" stored="true"/>

<field name="summary" type="text\_general" indexed="true" stored="true"/>

<field name="book\_format" type="text\_general" indexed="false" stored="true"/>

<field name="publisher" type="string" indexed="true" stored="true"/>

<field name="reviewer" type="text\_general" indexed="true" stored="true"/>

<field name="review\_text" type="text\_general" indexed="true" stored="true"/>

**<field name="icon\_url" type="text\_general" indexed="false" stored="true"/>**

<field name="image\_url" type="text\_general" indexed="false" stored="true"/>

<field name="audio\_url" type="text\_general" indexed="false" stored="true"/>

<field name="video\_url" type="text\_general" indexed="false" stored="true"/>

In addition, we also copied isbn, title, and author to the “text” field (default search field). Solr will be indexing and provide faster search result.

<copyField source="isbn" dest="text"/>

<copyField source="full\_title" dest="text"/>

<copyField source="author" dest="text"/>

<copyField source="summary" dest="text"/>

## Scheduling Import with DataImportScheduler

We created the a schedule task that runs every 60 minutes to import eXist-db xml files into solr. This is accomplished by creating dataimport.properties file under solr\_home/example/solr/conf folder

### Listing of dataimport.properties

# to sync or not to sync

# 1 - active; anything else - inactive

syncEnabled=1

# which cores to schedule

# in a multi-core environment you can decide which cores you want syncronized

# leave empty or comment it out if using single-core deployment

syncCores=coreHr,coreEn

server=localhost

port=8080

# application name/context

# [defaults to current ServletContextListener's context (app) name]

webapp=solr

# URL params [mandatory]

# remainder of URL

params=/select?qt=/existimport&command=full-import

# schedule interval

# number of minutes between two runs

# [defaults to 30 if empty]

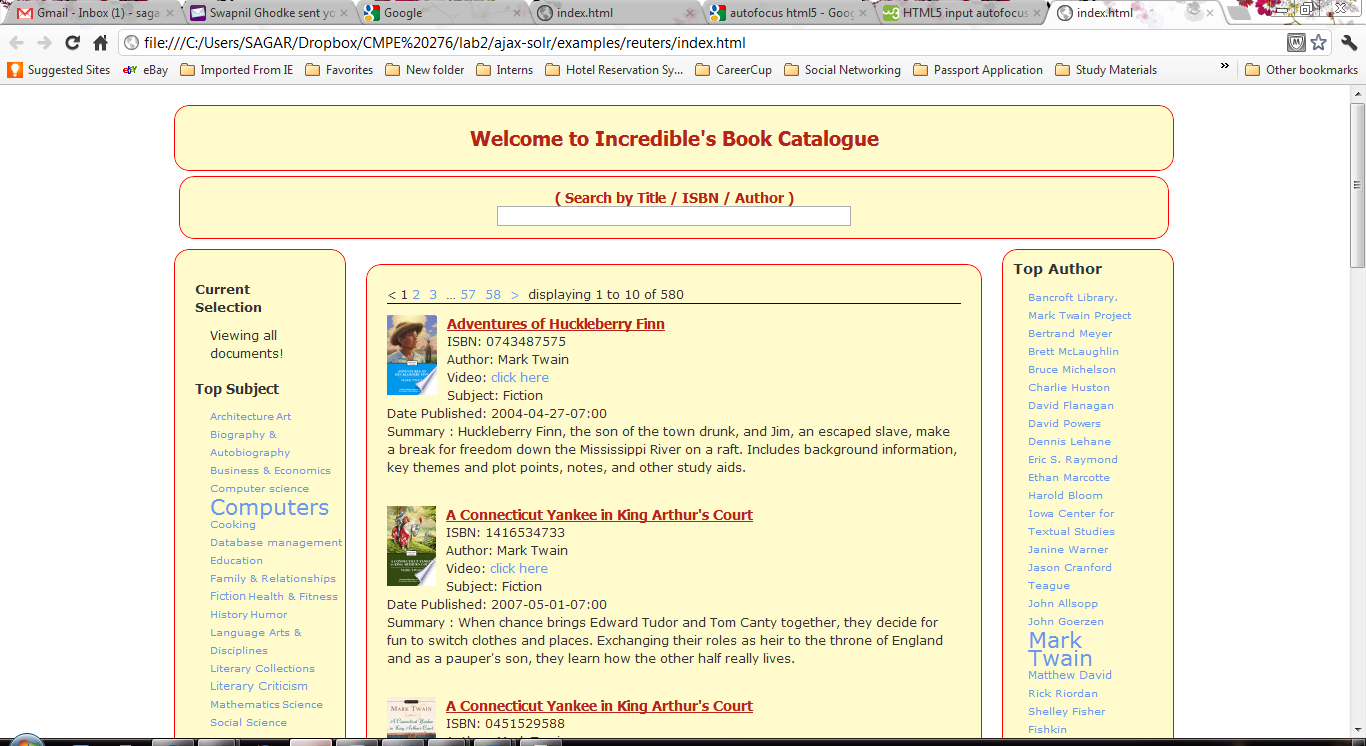
interval=60

# Features of On-Line Book Catalogue

Below are the screenshots of the Team Incredible Online Book Catalogue which self explains the usage:

## Home Page Navigation

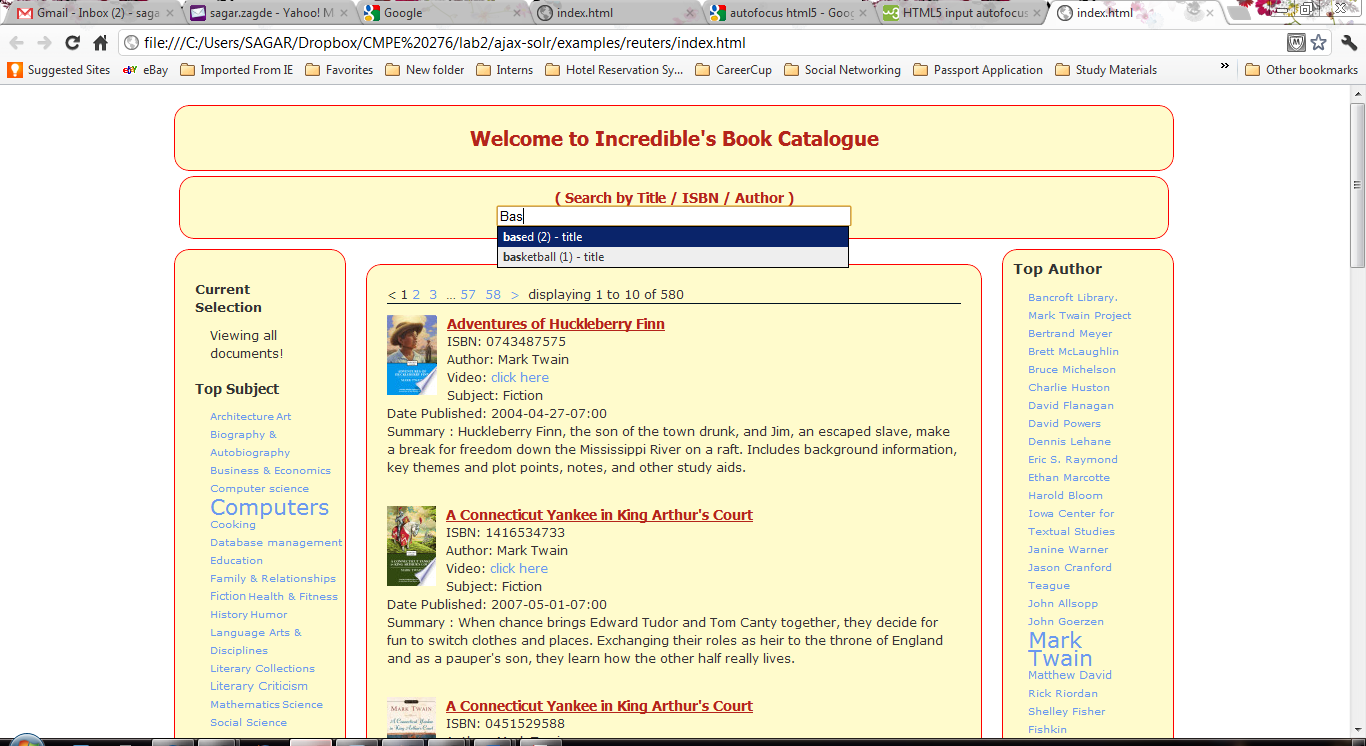
The user can see the books as shown below in center panel of result found. By default all the books will be displayed alphabetically. The two side panels can be seen. The search box is displayed which allows user to enter the Title or ISBN or Author to find books directly. As user tries to enter the Title, the hints for the Title according to the letter entered gets popped up in drop down select menu.



## Browse Books

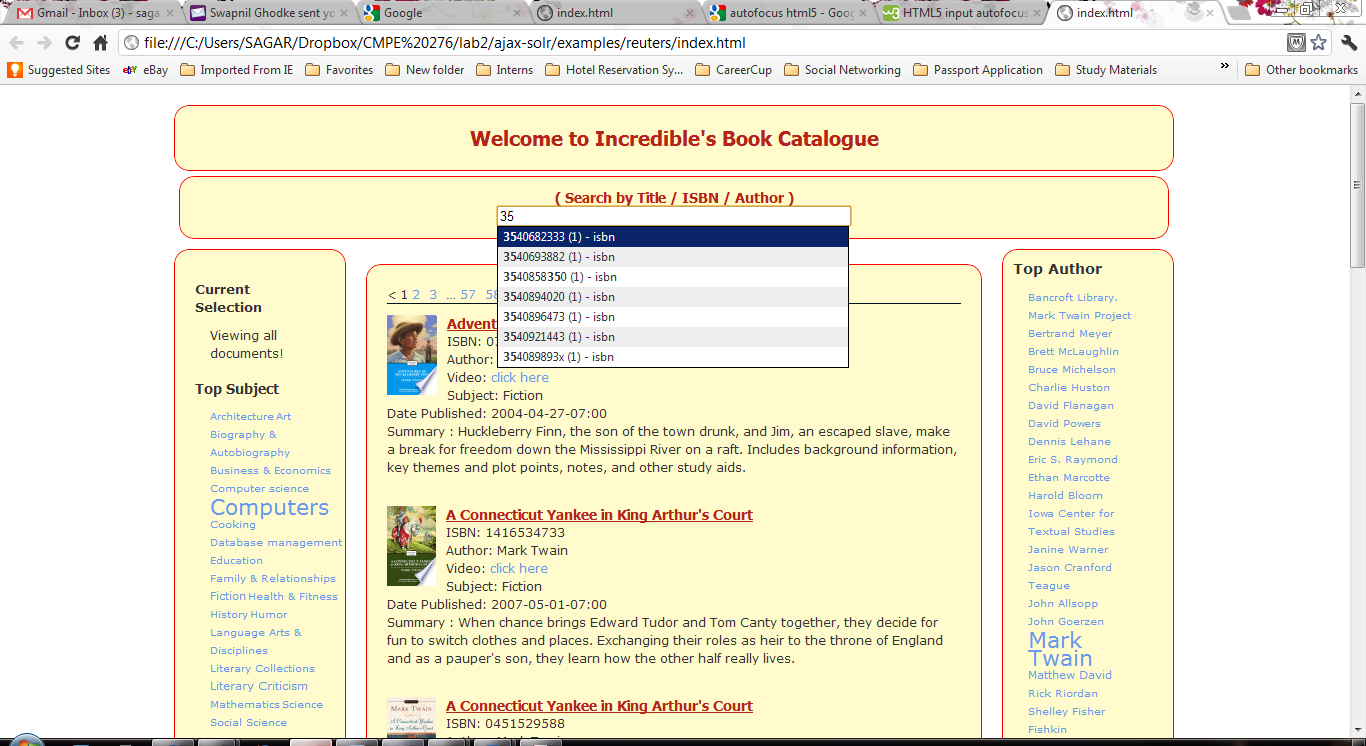
### By Title

The user can enter the Title of the book directly to search that book and the hints gets displayed for the title in the drop down menu to select from the menu and to search the books with that title.



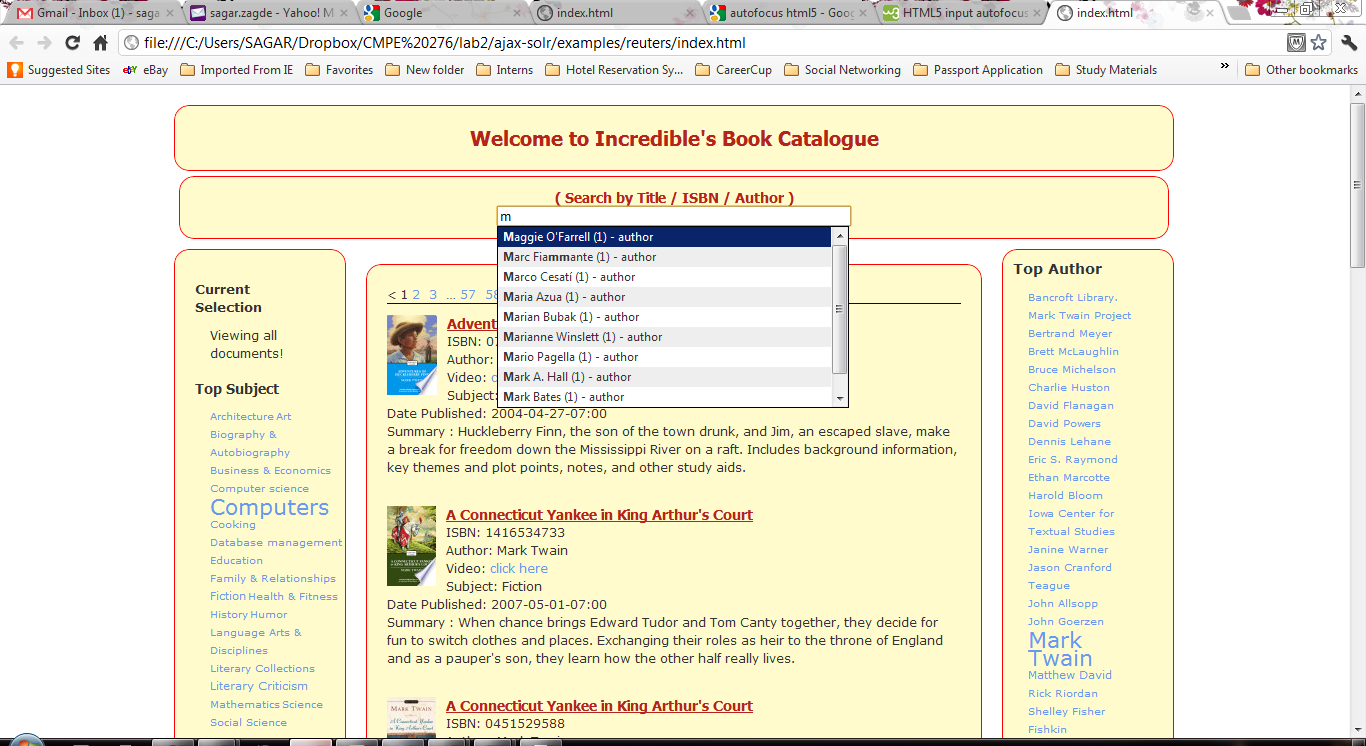
### By ISBN

The user can enter the ISBN of the book desired to search directly the book and the hints gets displayed for the ISBN in the drop down menu to select from the menu and to search that books with specific ISBN.



### By Author

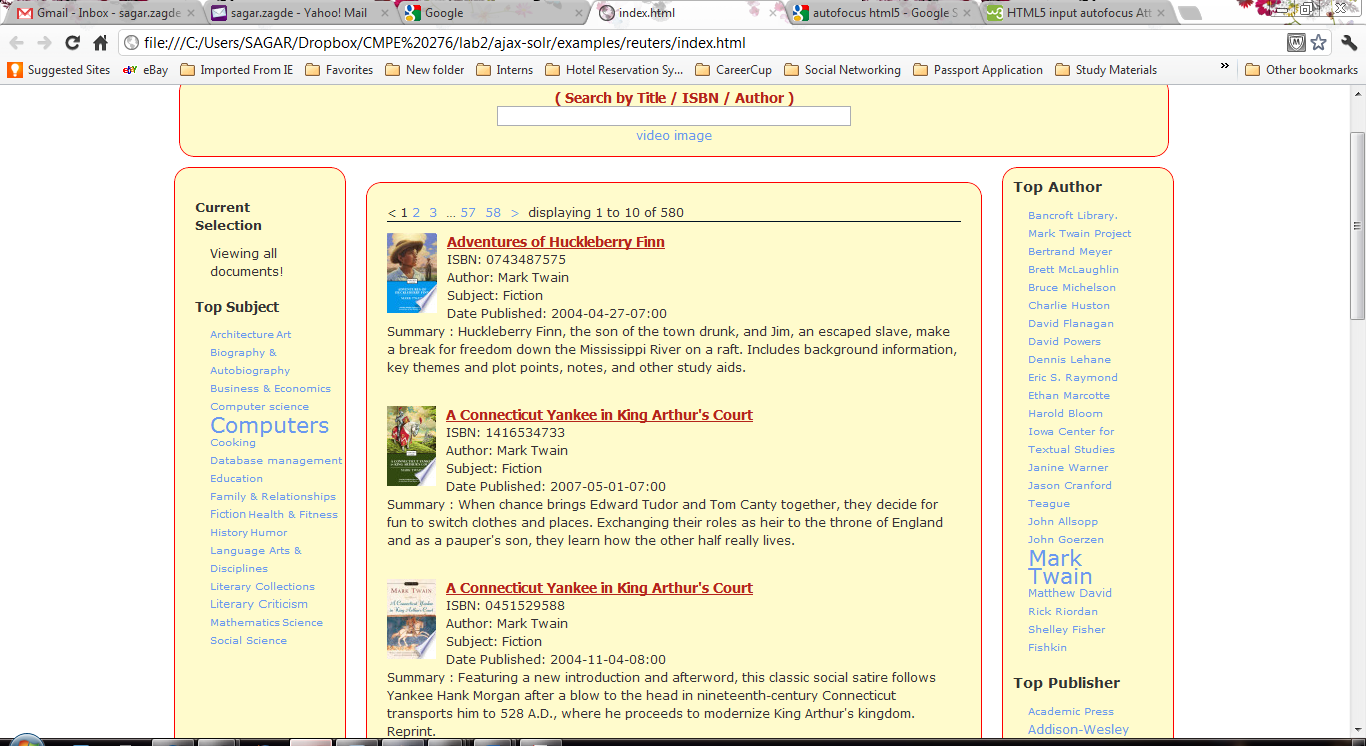
The user can enter the author name to search for the book and the hints gets displayed for the author name in the drop down menu to select from the menu and to search the books for that author.



## Aside Links

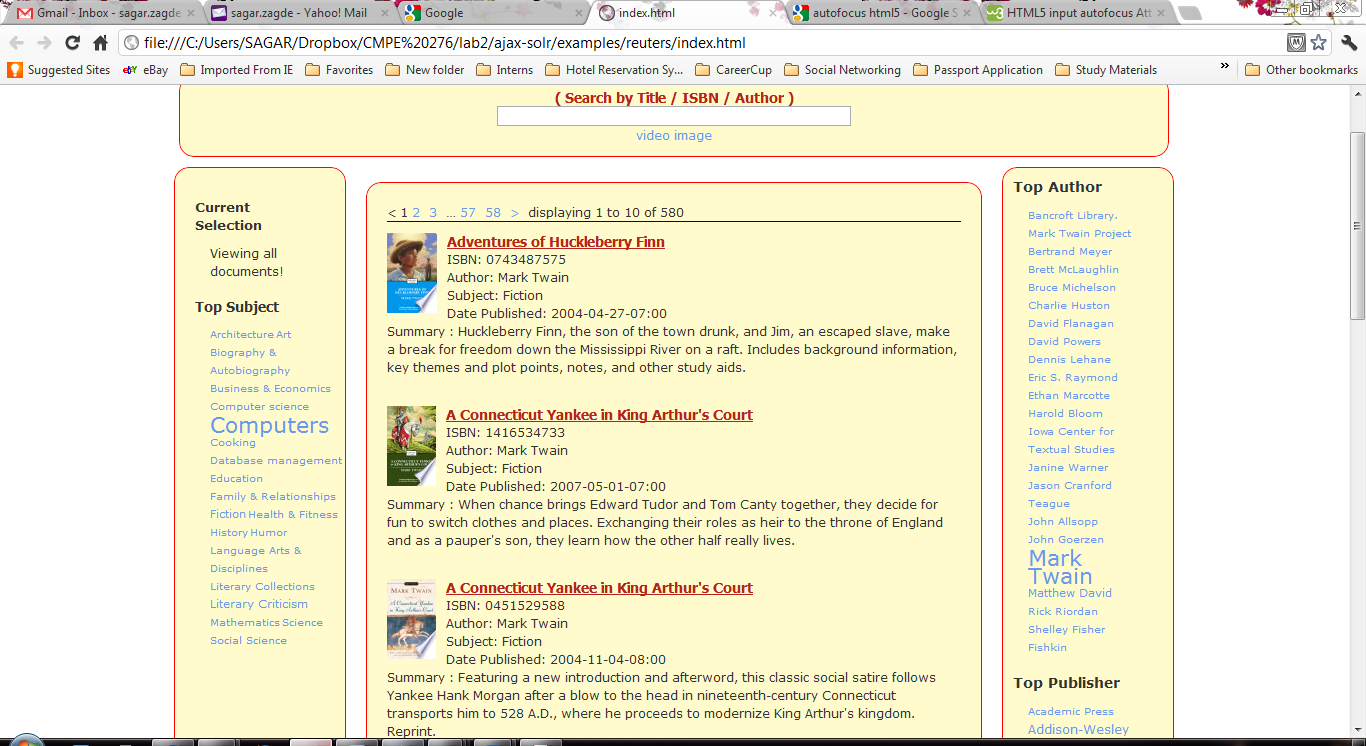
### Top Subject

The top subject links gets displayed below in the side panel as shown below. The subjects which has more than 100 entries will be shown in large letters as shown below for **Computers**



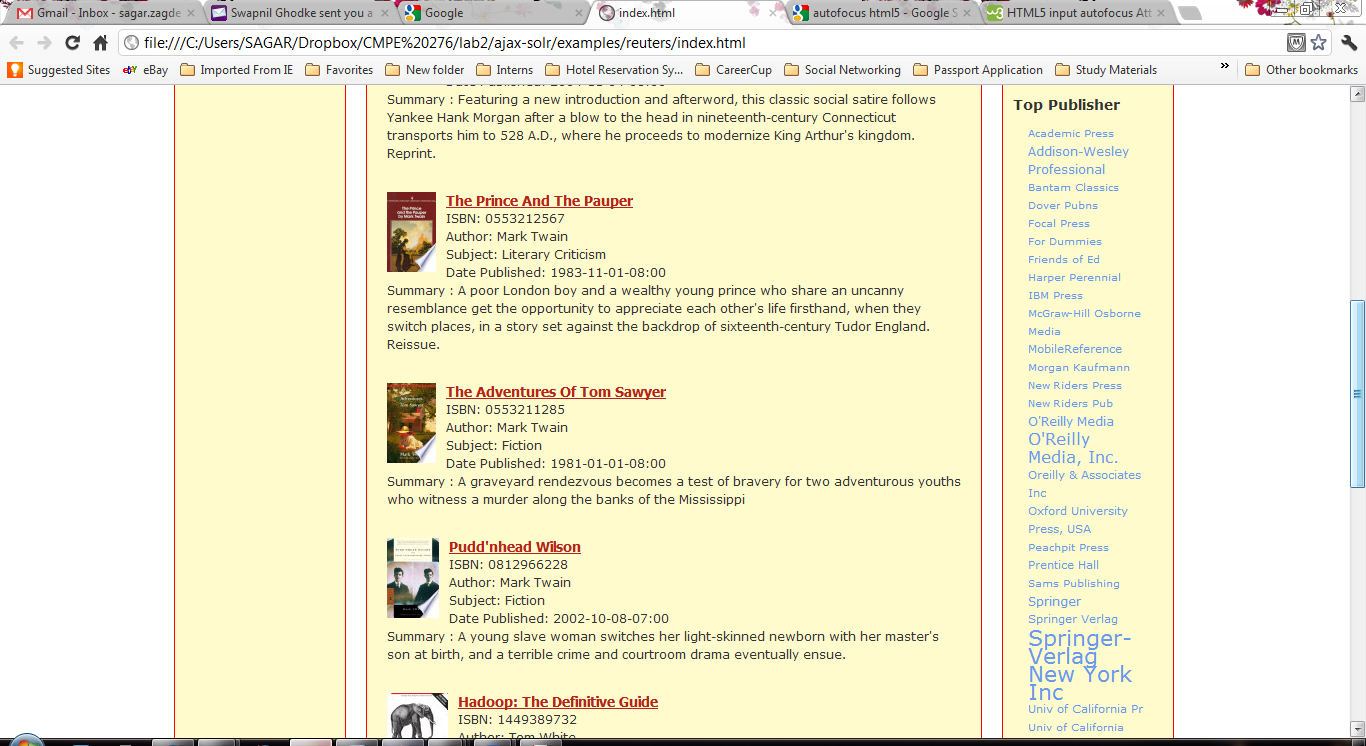
### Top Author

The top authors’ name gets displayed below in the right side panel as shown below. The authors who have written more than 100 books their names will be shown in large letters as shown below for **Mark Twain**



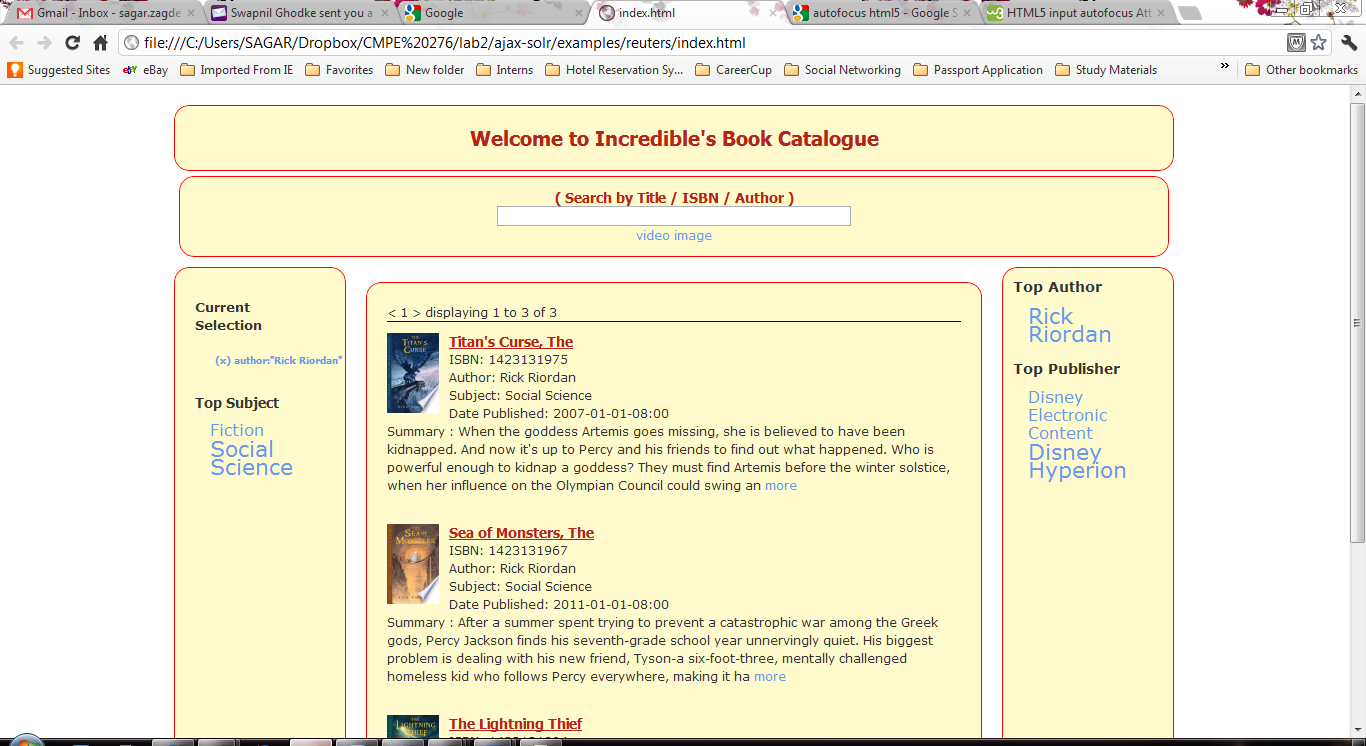
### Top Publisher

The top publishers’ name gets displayed below in the right side panel below Top author as shown in below screenshot. The publishers who have published more than 100 books their names will be shown in large letters as shown below for **Springler-Verlag New York Inc.**



### Current Selection

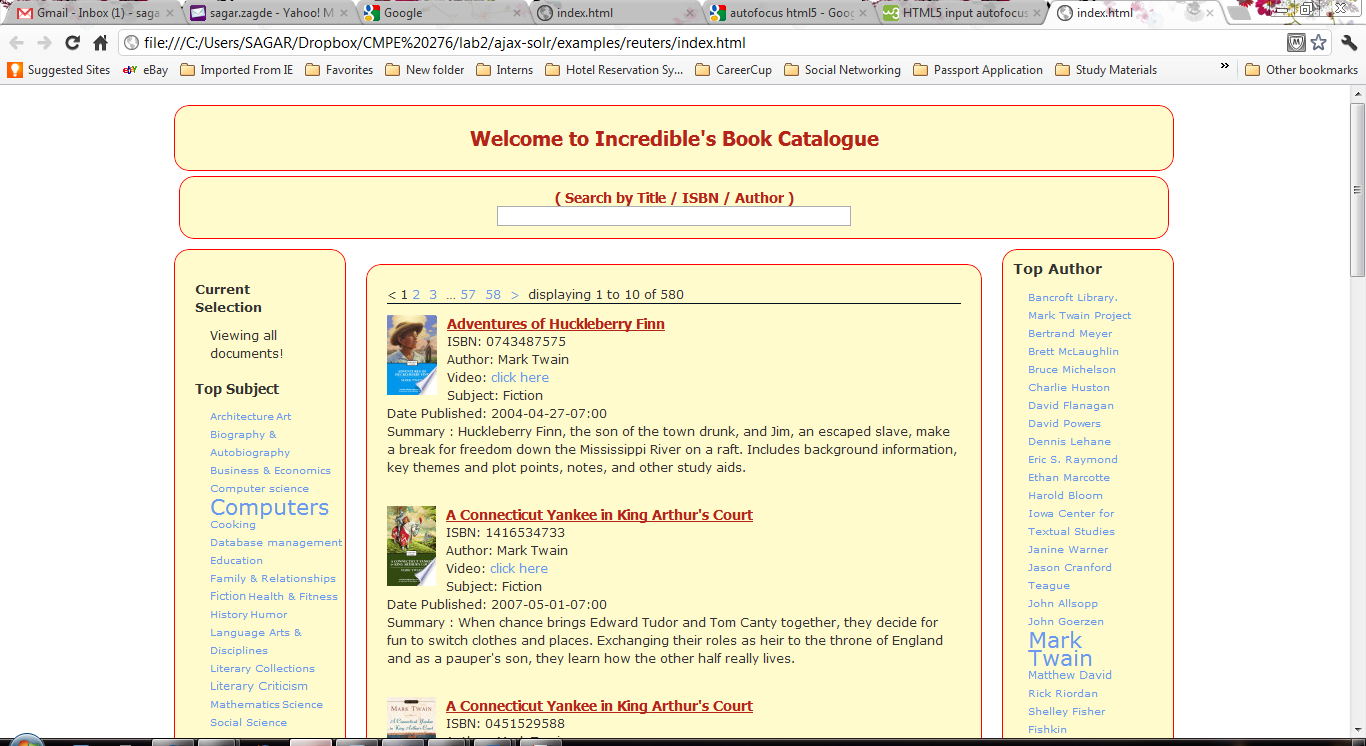
The books selected Subject or Author or Publisher category will be displayed in this panel on left top side as shown below. As shown the Rick Riordan author is selected and all the books written by him gets displayed in the display center panel as highlighted.



## Navigation of Pages and Summary related feature

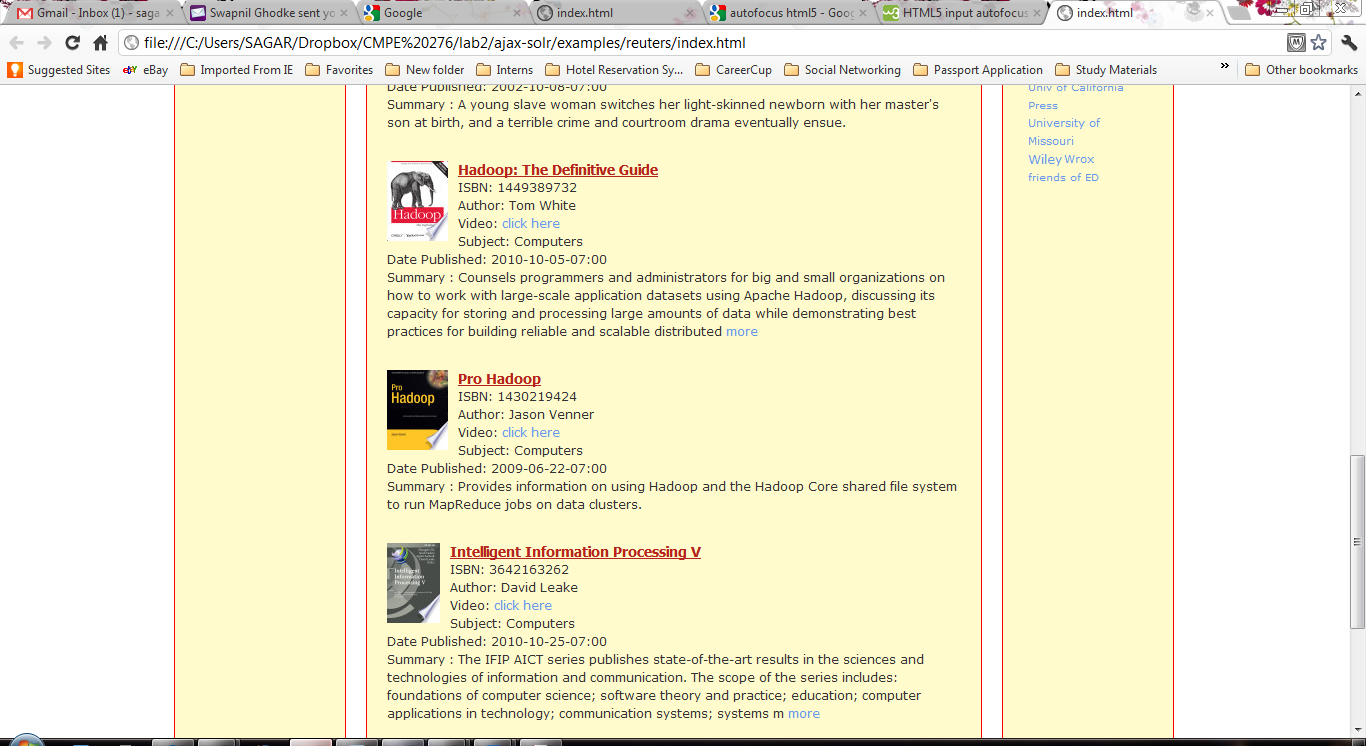
### Navigation through pages for search

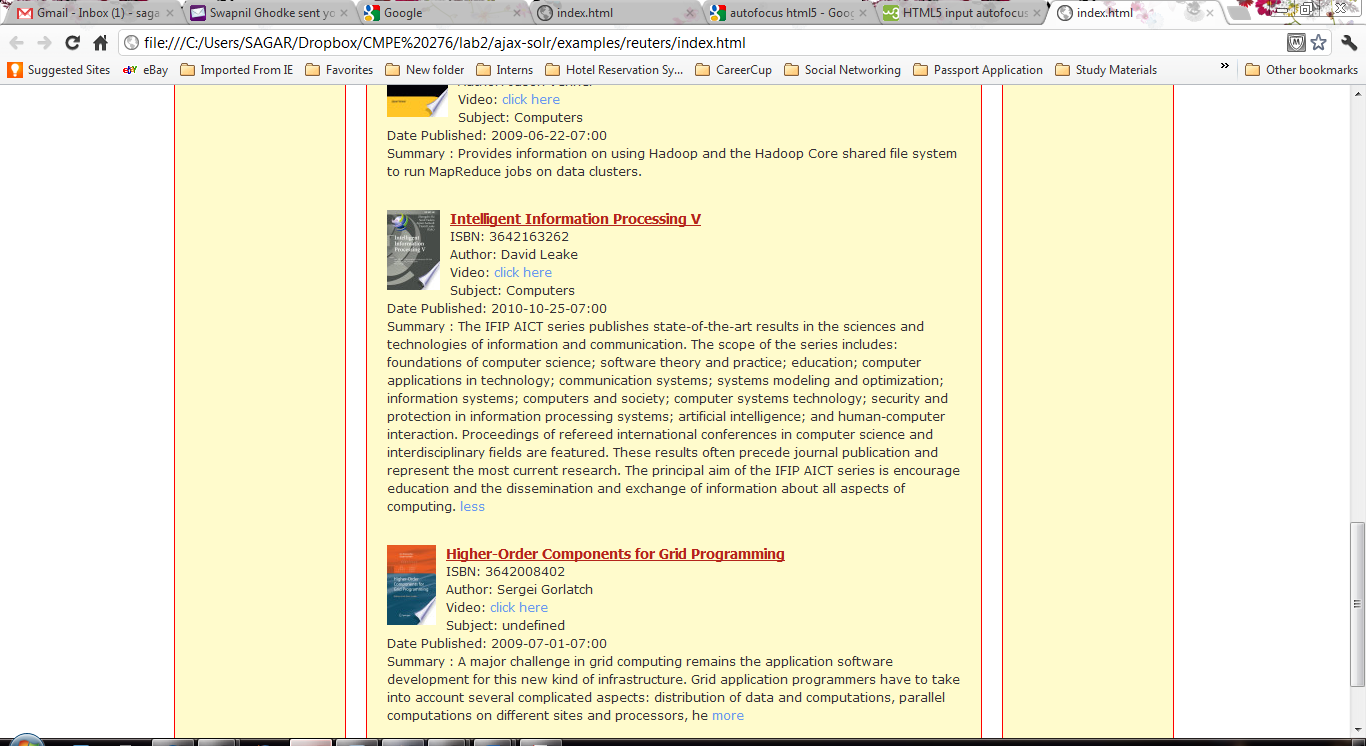
The user can navigate through the search results found for the selection. The page numbers will be displayed as shown in below screenshot. By page numbers the user can jump directly to the specific page to display. The page displayed will be shown in black and the other pages which can be navigated get displayed in hyperlinks in blue which can be clicked to navigate to that page. Below shows the 580 pages found of containing book details but only 58 are limited to display.



### The Summary feature of Books

If the summary of the book found to be more than 300 letters than the “more” as a hyperlink will be displayed so that user can expand if he desires to. And when expanded the user will see a “less” as a hyperlink show that user can minimize the summary greater than 300 letters. See the below screenshot.

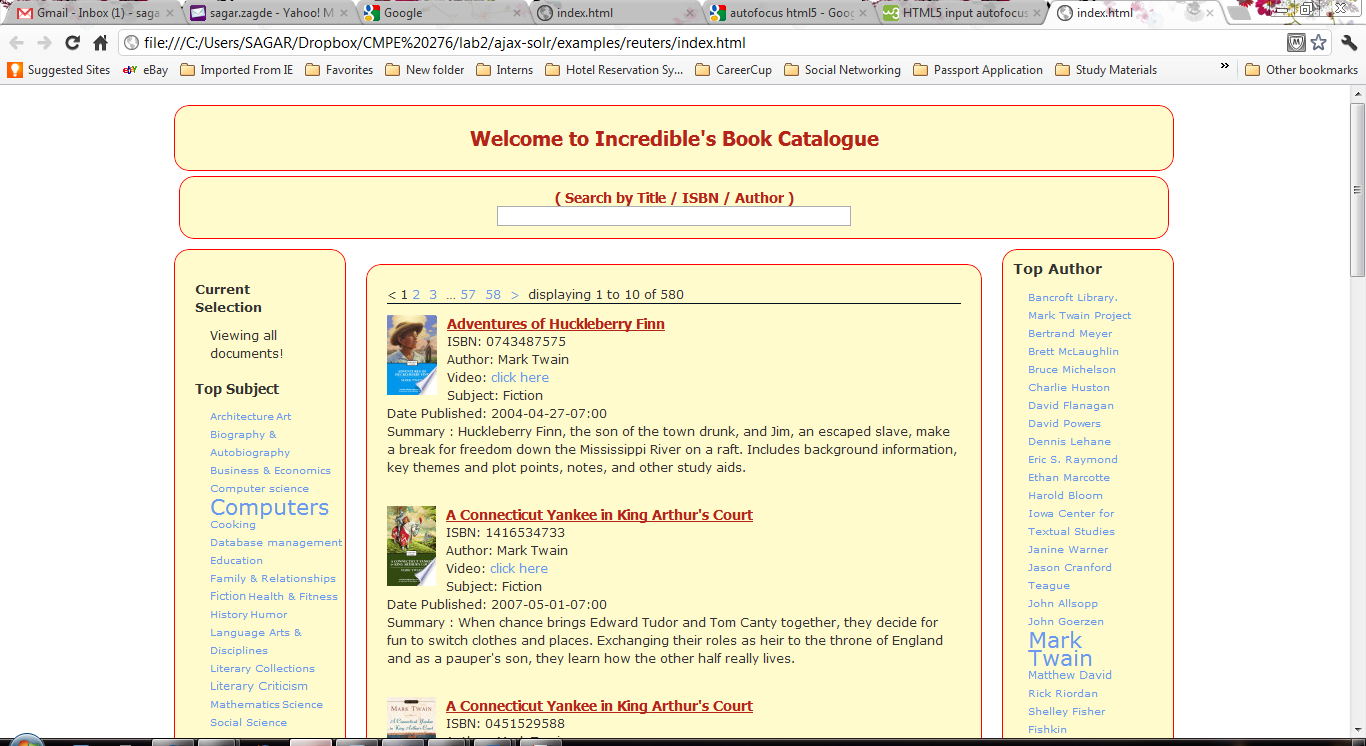


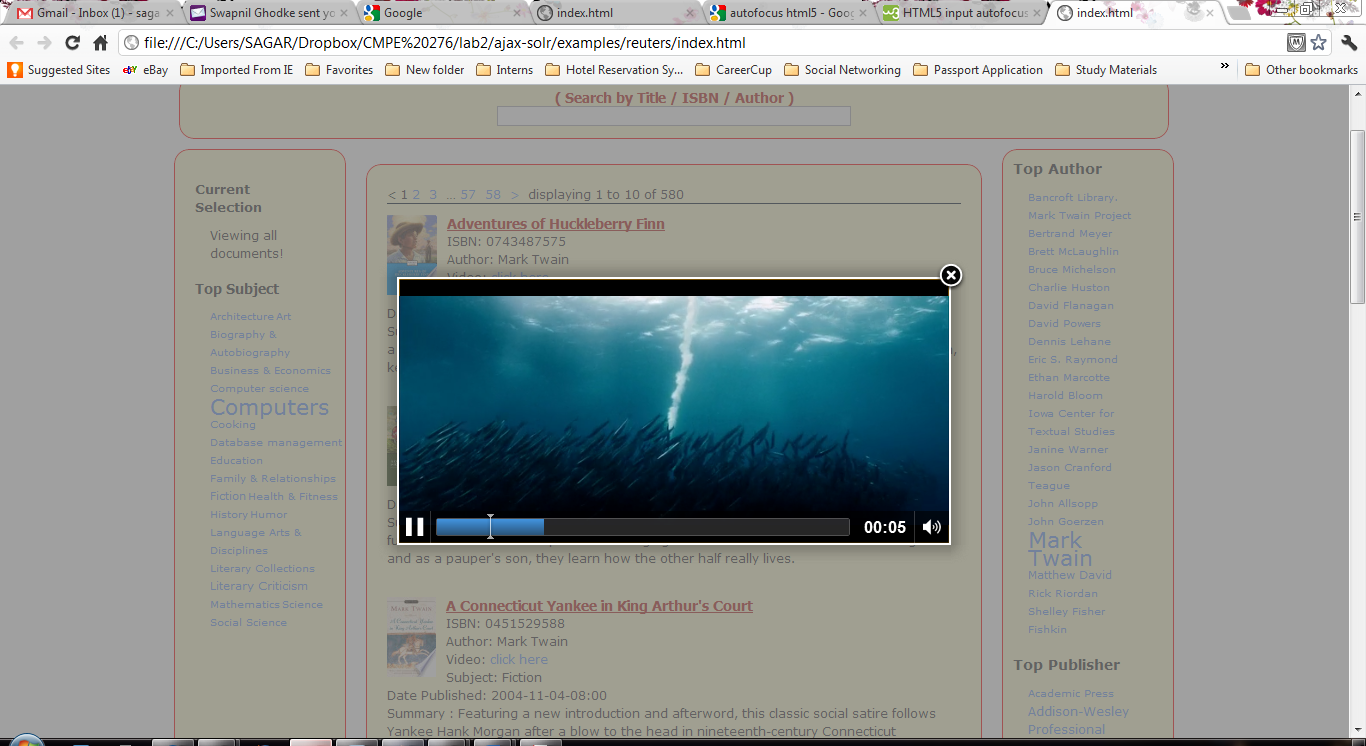


# HTML5 and CSS3 Features used

1. Creating pop-up Windows with Custom Data Attributes
   1. **Video of book to play**

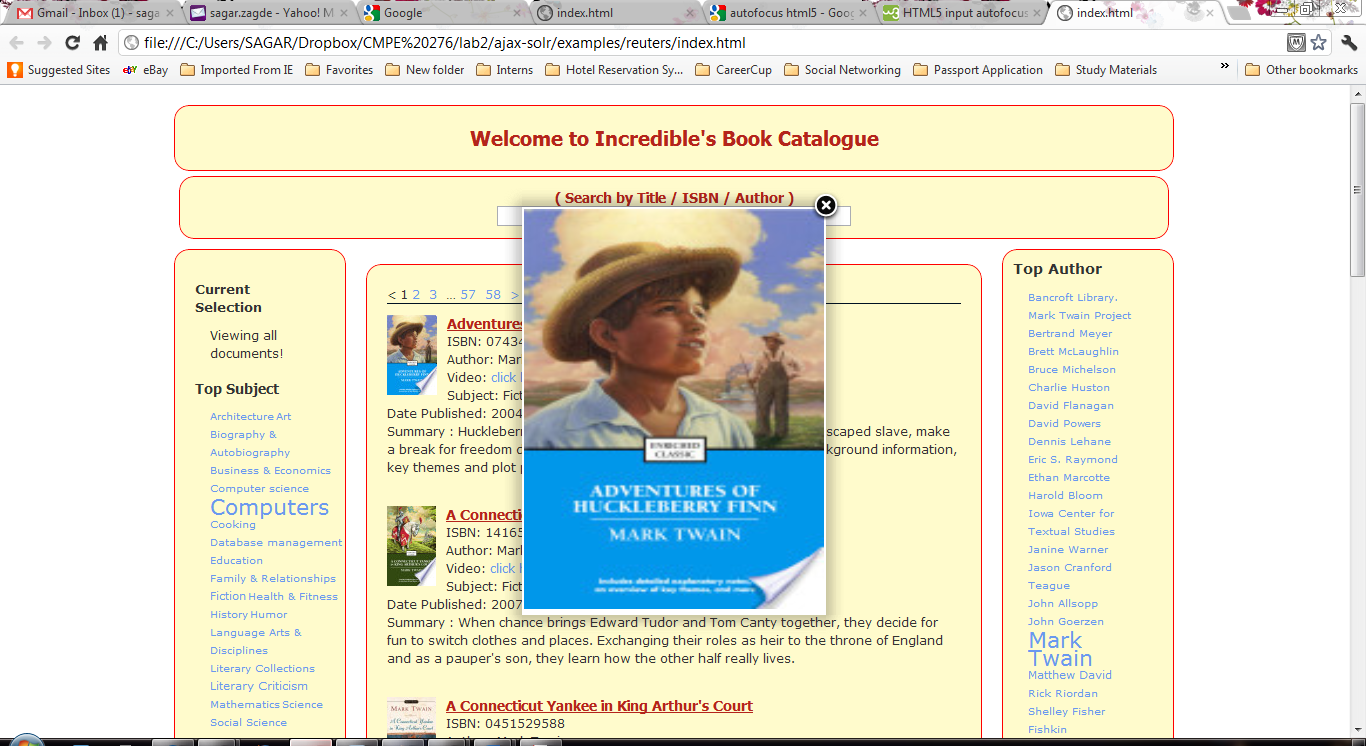
When the user click on the link shown by “click here” hyperlink, the video panel opes as shown in below screenshots





1. Autofocus: Jump to the first field on forms or search fields
2. Creating multi-column layouts
3. Canvas feature due to which user can have a larger image to look at.

When the user click at the image the image gets enlarged as shown in below screenshot

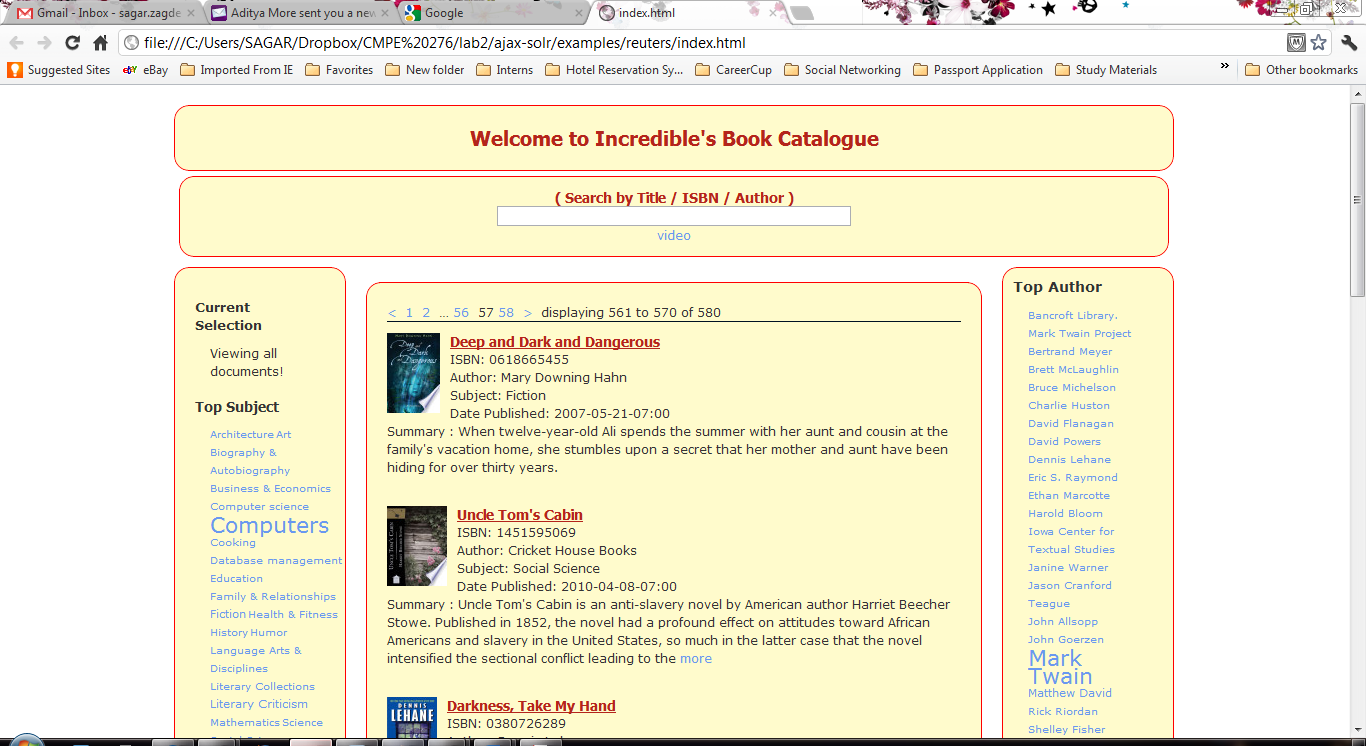


1. Use of new Semantic/Structural tags

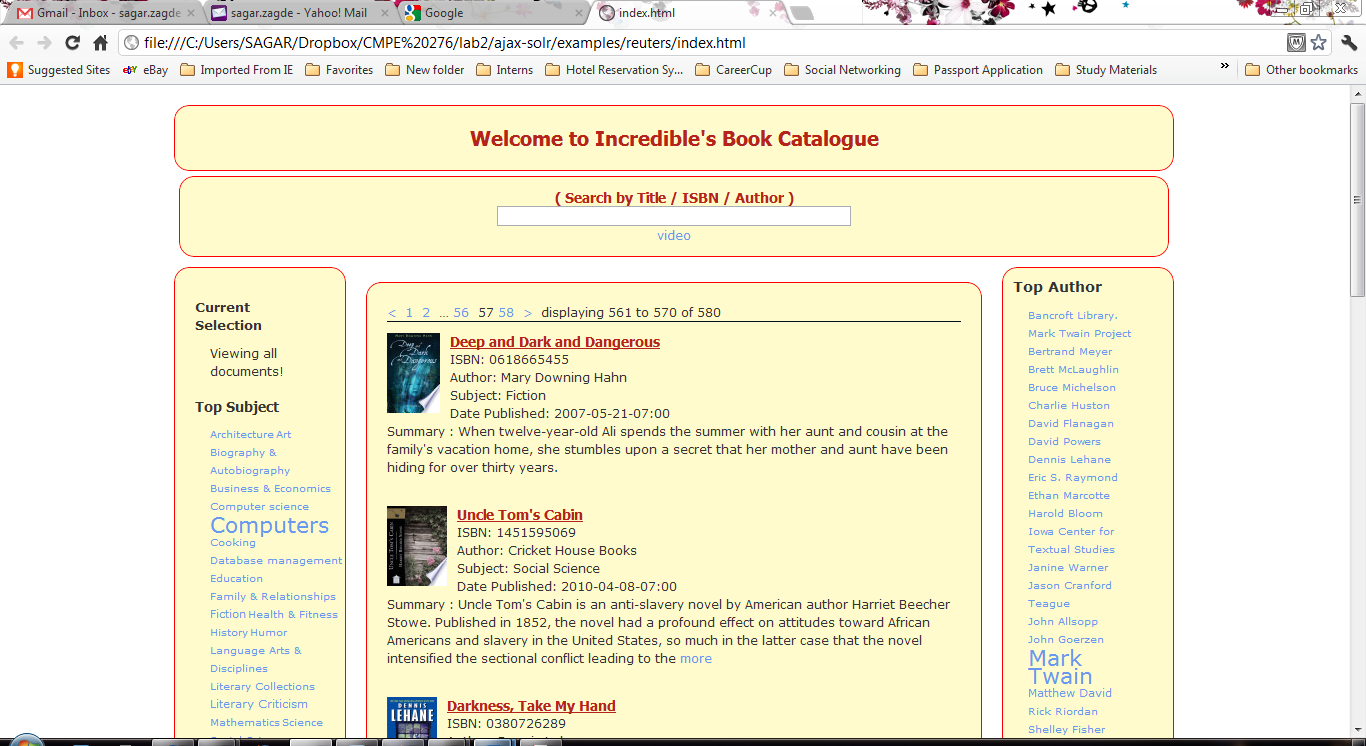
<source>, <nav>, <section>, <article>, <aside>, <video>, <audio>, <article>, <header>, <footer>

1. Use of new features of CSS3
2. Rounding Rough edges for search display panels done by CSS3

-webkit-border-radius: 15px



1. webkit usage : -webkit-box-orient : horizontal



1. -webkit-box-flex : 1;

As seen above we have kept the size of <aside> fixed and the center section auto arranges itself as needed

1 🡪 auto arrange; 0🡪 fixed model

# Problem & Enhancement

## Problem with XPathEntityProcessor

### Description

For our project, since we’re relying on eXist-DB RESTful interface to get to the XML files, we must create two XPathEntityProcessor. The parent <entity> node will get a list of all the xml files to process in the eXist-DB collection. The nested child <entity> node then parses each file to retrieve the relevant data. Since the child XPathEntityProcessor was already initialized in the parent node, it could not create one document object per child record properly. The problem has a Jira defect logged on Apache here:

<https://issues.apache.org/jira/browse/SOLR-2094>

### Work Around

Since we’re using Google Book API and JAX-B we have complete control over our XML file generation process. We coded our program such that it will create one file per book record.

### Future Improvement

We’re considering if it’s possible to fix the defect & submit the fix to Apache.

## XML Importer does not have incremental import

### Description

The problem here is that even though the DataImportHandler has some facility built in for delta import; eXist-DB has the timestamp for when the file was stored; we cannot use delta import because the XPathEntityProcessor does not support it.

### Work Around

No work around currently. We are doing full import everything the collection changes.

### Future Improvement

We’re considering if it’s possible to extend XPathEntityProcessor to specifically work with eXist-DB.

# Conclusion

In this lab, a web based application that uses xml documents to represents an online book catalog was successfully designed and implemented. Users will be able to access the web page through any browser and search for any book available inside server’s database. The site will allow user to sort books by category, author, or even publisher. Through this lab, we learned how to use eXist and Solr to store and search from within the database. In addition, we became even more comfortable using HTML5 to design web based graphical user interface. If time was not a factor, some additional features that we could implement to enhance this online book catalog would be a user login, online audio book reading and book video. These features will not only attract more users but it will allow user to become more interactive with the book of their desire.

# References

[1] J. Dyer. (2012, March 22). *Apache Solr Wiki*. [Online]. Available: <http://wiki.apache.org/solr/DataImportHandler#Data_Import_Request_Handler>

[2] W. M. Meier. (2010, March 1). *eXist Quick Start*. [Online]. Available: <http://exist-db.org/exist/quickstart.xml>

asdf

[3] JAX-B Tutorial. Retrieve on 4/4/2012: <http://jaxb.java.net/tutorial/section_1_3-Hello-World.html>