International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 4 Issue: 6 404 - 411

JSP Custom Tag for Pagination, Sorting and Filtering – A Case Study

Dr.Poornima G. Naik
Professor
Department of Computer Studies
CSIBER
Kolhapur, India
pgnaik@siberindia.edu.in

Girish R. Naik
Associate Professor
Production Department
KIT's College of Engineering
Kolhapur, India
girishnaik2025@gmail.com

Abstract— Tag libraries have the power of reducing complex functionalities to one liners by separating out implementation part from tag declarations. Tags do hide the implementation specific tasks from the end user by making the code more readable. The frequently and widely used functionality in any application is database operations which involve lot of code repetition. Such a repeated code can be hidden behind a couple of custom tags where the end user can be concerned only with the tag usage which renders the application bug free and also aids in rapid application development. Majority of automation softwares at the minimal incorporate functionalities for interaction with repository of data . The need for quick searching of required data and retrieving subsets of data demand sorting, pagination, and filtering capabilities to be an integral part of any application. With the exponential growth in data these functionalities become mandatory to be incorporated in any application irrespective of its type and size. Further, Rich Internet Application (RIA) demands an attractive graphical user interface providing visual clues on the type of data to be entered or to be displayed. In order to cater a solution to this issue, in the current paper, the authors have designed and implemented a JSP custom tag for displaying a database table data in columns of different types such as check boxes, images, hyperlinks etc. Boolean attributes are added to the tag for enabling one or more of the features corresponding to pagination, sorting and filtering.

Keywords- *BodyTagSupport, Graphical User Interface, JSP Custom Tag, Tag Attribute, Tag Library Descriptor, Tag Handler Class.*

*****

I. INTRODUCTION

When displaying huge data, it is often desirable to display only a portion of the data at the outset and then allow the user to step through the data a specified no. of records at a time. At the same time an end user's experience can be enhanced if they are able to view data sorted by one of the columns. Incorporating these functionalities in a pair of custom tags offers several advantages as the tags provide implementation of these functionalities in a platform independent way and operate in a secure environment. One of the authors has demonstrated implementation of JSP custom tags for displaying the contents of table for variety of backend database management systems, performing various DML operations on database and displaying master-detail relationships by encapsulating the large amount of JDBC code behind couple of custom tags. In the current work, authors demonstrate incorporating sorting, pagination and filtering capabilities in custom tags.

The prime benefits offered by custom tags are two fold. On one hand they play a key role in realizing code reusability and on the other hand enable code separation from its presentation. Since the tags are based on the proven Java technology, they reap the benefits offered by the Java technology such as security, robustness and platform independency apart from code reusability.

The preliminary steps employed in implementation of a custom tag are [1] :

- Implementation of a tag handler class
- Generation of a tag library descriptor document for storing tag specific information in XML file format.
- Designing JSP page using custom tag.

II. LITERATURE REVIEW

Custom tags play an important role in web applications. JSP custom tags are written to extract data from database using drop down menu to generate options dynamically [4]. A through investigation for categorization of requirements and design of tag software in web application has been carried out by [5]. Authors have presented a case study of freely available tag software. The development and testing of an accurate mass–time (AMT) tag approach for the LC/MS-based identification of plant natural products in complex extracts has been reported by [6]. Its utility is verified by the detection and annotation of active principles in different medicinal plant species with diverse chemical constituents. Tagging plays a vital role in bioinformatics also . A method to generate poly(A) tags libraries for high-throughput sequencing (PAT-seq) has been reported by [7]. This method has been applied to investigate mRNA polyadenylation in Arabidopsis. Internet has become a vital source of information. Due to this there is need for powerful internet systems which can help in audiovisual content searching on internet. A new technique of searching and indexing of audio visual contents on the internet has been carried out by [8]. When developers are working on different platforms then code migration is a major issue. Three methods of code migrations from JSP to ASP.NET Entire code transform migration, Reserved migration and Neutral migration has been proposed by [9]. In development of IOT based applications there is need for a way to connect things and services together and processing of data emitted by them using data flow paradigms. Automation of distribution of these data flows using appropriate distribution mechanism has been carried out by [10].

III. Theoretical Framework for Tag Design

Since the tag works independent of back end database management system, retrieving a range of records sequentially poses a big challenge in implementation. Further, MS-Access does not support any pseudo columns as is the case with other back end database managements systems such as MySQL , Oracle etc. These systems support pseudo columns with the name either ROWID, or ROWNUM, depending on the system which makes the task easier. The authors have come up with a query in MS-Access for sequential retrieval of records in a specified range.
For retrieving first N records from book table, the MS-Access query is

SELECT TOP N * FROM book;

For retrieving the records in the range N1-N2 the query can be formulated as shown below:

No of Records per Page = (N2-N1+1)

SELECT TOP <No of Records per Page> * FROM book WHERE <PK> not in (select TOP <N1-1> * No of Records per Page > <PK> from book);

where <PK> refers to primary key column. Table I. depicts using the query for retrieving records in different range.

TABLE I. Retrieving Records in Different Ranges in MS-Access.

Page No	Range	Query
1	1-5	SELECT TOP 5 bookid FROM book;
2	6-10	SELECT TOP 5 bookid FROM book WHERE bookid not in (select TOP 5 bookid from book);
3	11-15	SELECT TOP 5 bookid FROM book WHERE bookid not in (select TOP 10 bookid from book);
4	16-20	SELECT TOP 5 bookid FROM book WHERE bookid not in (select TOP 15 bookid from book);
5	21-25	SELECT TOP 5 bookid FROM book WHERE bookid not in (select TOP 20 bookid from book);

A. Tag Library Descriptor Document

A tag library descriptor file is a simple XML file with the extension .tld embedding a set of custom tags. The structure of a typical tag library descriptor file is shown below:

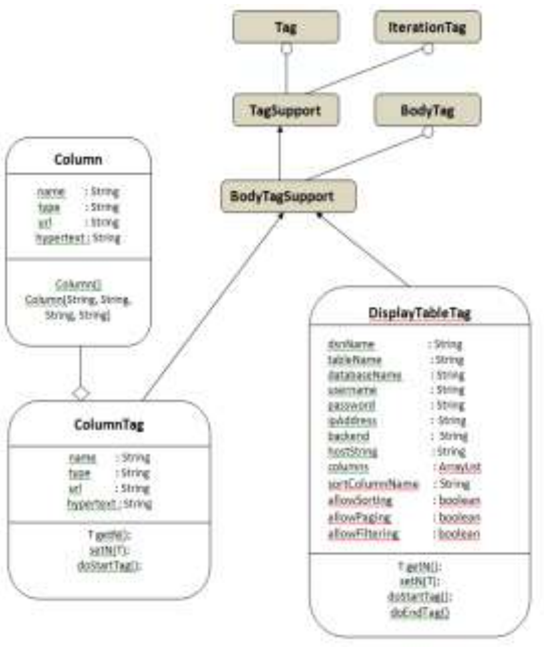
```
<?xml version="1.0" ?>
<!DOCTYPE taglib
  PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.2//EN"
  "http://java.sun.com/dtd/web-jsptaglibrary_1_2.dtd">
 <taglib>
  <tlib-version>1.0</tlib-version>
  <jsp-version>1.1</jsp-version>
  <short-name>simpletaglib</short-name>
  <description>My first Tag Library</description>
<tag>
  <name>…</name>
  <tag-class>…</tag-class>
  <body-content>…</body-content>
  <attribute>
   <name>…</name>
    .
    .
  </attribute>
   .
   .


 </tag>
 .
 .
 <tag>
 </tag>
</taglib>
```

The required child elements of <tag> element are <name>, <tag-class> and <body-content> and optional child element is <attribute>. The <attribute> child element contains the compulsory child element <name> and other optional child elements such as <rtexprvalue>, <required>, etc.

B. Class Diagram

The structure of the various classes employed in tag design and interaction between them is depicted in Figure 1.



1

Figure 1. Structure and Relationship Between Tag Handler Classes

The classes with solid background are pre-defined classes while those with transparent background are custom classes.
The type attribute of a column tag can take one of the following values.

- regular
- checkbox
- image

- hyperlink

The url and hyperlink attributes are NOT NULL only if the type attribute is set to hyperlink while the url attribute alone is NOT NULL if type attribute is set to image. This situation is depicted in Table II.

TABLE II VALIDATION OF ATTRIBUTES OF COLUMN TAG EMPLOYED IN TAG DESIGN

	url	hypertext
Regular	X	X
checkbox	X	X
image	✓	X
hyperlink	✓	✓

The life cycle of a tag handler class consists of 6 methods of which doStartTag() and doEndTag() are frequently implemented[11,12]. doStartTag() method is called when the start tag is rendered which returns one of the constants EVAL_BODY_INCLUDE or SKIP_BODY. In the former case the body content of the tag which consists of nested column tags in our case is evaluated while in the latter case the last method in the life cycle of the tag doEndTag() is called. The functionalities offered by the various tag handler classes are shown in Table III.

TABLE III FUNCTIONALITIES ASSOCIATED WITH TAG HANDLER CLASSES

Tag Handler Class	Method	Functionality	Return Type
DisplayTableTag	doStartTag()	Creates an ArrayList for storing column specific information.	EVAL_BODY_INCLUDE
	ArrayList columns = new ArrayList(); pageAttribute.setAttribute("columns",columns);		
	doEndTag()	Retrieves column information stored in columns of type ArrayList and renders them appropriately in different columns of a table.	super.doEndTag()
	ArrayList columns =(ArrayList)pageContext.getAttribute("columns");		
ColumnTag	doStartTag()	Retrieves ArrayList storing column specific information from current session, appends the current column information to it and writes it back to the current session.	SKIP_BODY

IV. IMPLEMENTATION OF CUSTOM TAG

This section presents structure of tag library descriptor file, the control flow diagram and the proposed algorithm for the implementation of a custom tag.

A. Writing a Tag Library Descriptor

DisplayTable tag contains the following attributes for adding sorting, pagination and filtering functionalities to the tag which are optional and are set to the default value of false.

- allowSorting
- allowPaging
- allowFiltering
- recordsPerPage.

The first three attributes have default value of false which means by default sorting, pagination and filtering are disabled. If pagination is enabled, and recordsPerPage is not specified, then recordsPerPage attribute defaults to a value of 10 records per page.

The following code segment shows only a partial content of customtag.tld file where the changes are incorporated.

```
<tag>
  <name>DisplayTable</name>
  <tag-class>csiber.DisplayTableTag</tag-class>
  <body-content>empty</body-content>
  <attribute>
    <name>dsnName</name>
  </attribute>
  <attribute>
    <name>tableName</name>
  </attribute>
  <attribute>
    <name>columnNames</name>
  </attribute>
  <attribute>
    <name>sortColumnName</name>
  </attribute>
  <attribute>
    <name>databaseName</name>
  </attribute>
  <attribute>
    <name>userName</name>
  </attribute>
  <attribute>
    <name>password</name>
  </attribute>
  <attribute>
    <name>backEnd</name>
  </attribute>
  <attribute>
    <name>allowSorting</name>
  </attribute>
  <attribute>
    <name>allowPaging</name>
  </attribute>
  <attribute>
    <name>recordsPerPage</name>
  </attribute>
  <attribute>
    <name>allowFiltering</name>
  </attribute>
 </tag>
```

Structure of ColumnTag Tag.

```
<tag>
<name>Column</name>
<tag-class>csiber.ColumnTag</tag-class>
<body-content>empty</body-content>
<attribute>
<name>name</name>
</attribute>
<attribute>
<name>type</name>
```

```
</attribute>
<attribute>
<name>url</name>
</attribute>
<attribute>
<name>hyperText</name>
</attribute>
</tag>
```

B. *Nesting Tags*

The column tag is nested inside DisplayTable tag. All the nested columns are displayed by DisplayTable tag in the specified column format. This is achieved by iteratively invoking column tag and storing column-specific information in a collection class of type ArrayList.

C. *Folder Structure of a Custom Tag Project*

Different type of project components in an Eclipse project folder with the name customtag is depicted in Figure 2.

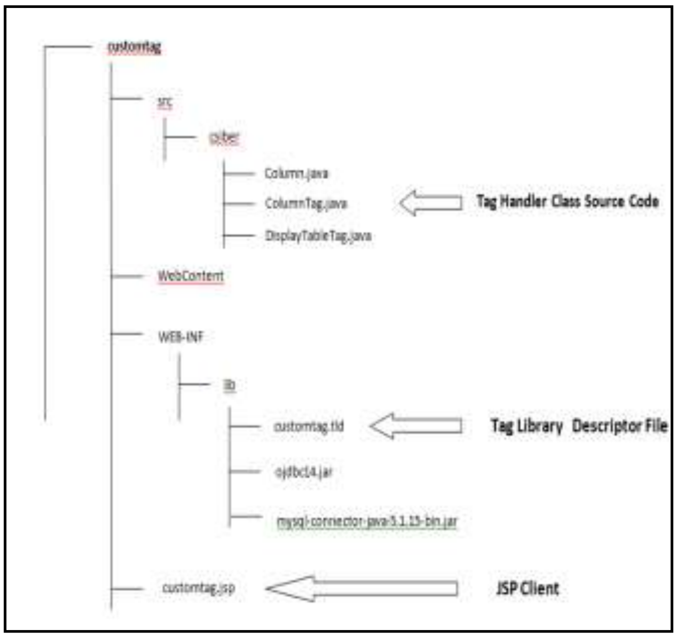


Figure 2. Folder Structure Employed in Implementation of Custom Tag

D. *Proposed Algorithm*

The algorithm for displaying the master-detail relationship in a hierarchical grid control in C++ style is presented below:

/*Any high level language interfacing with back end database management system provides high level API for primitive database functions such as creating a connection object and generating a page request by sending the necessary input information in a query string. Hence this algorithm assumes some standard functions as shown below:

Standard Functions of language L used in the Algorithm

loadDriver() - is function in a language L for loading appropriate DBMS driver in memory depending on the name of DBMS passed as parameter.

connectTo() - is a function in a language L for establishing the connection to remote DBMS depending on the name of DBMS passed as a parameter.

getPageName() - is a function in language L for returning the name of the web page requested.

getQueryString() - is a function in language L for returning the value of the query string parameter whose name is passed as a parameter to the function.

constructQuery() – is a function in language L for constructing an SQL query for pulling data from the table whose name is passed as a parameter.

executeQuery() – is a function in language L for executing the query against backend database management system.

startsWith() - is a function in language L for checking whether the string passed as a first parameter starts with the character passed as second parameter.

displayDetailRecords() - is a function in language L for displaying records of a detail table.

```
/*
Global Variables
String data;
ArrayList columns;
String query;
*/
struct Column
{
    String name;
    String type;
    String url;
    String ctype;
    String hyperText;
}

/* Invoked when start tag is rendered */

public int doStartTag()
{
            data=null;
            columns=new ArrayList();
            pageContext.setAttribute("columns", columns);
            return EVAL_BODY_INCLUDE;
}

/* Invoked when end tag is rendered */

public int doEndTag()
```

```
{
            /* Retrieve column information */
            String columnValue=null;
            columns=(ArrayList)pageContext.getAttribute(
                                                    "columns");
        /* Retrieving Page name and query string information */
                String pagename =
              this.pageContext.getPage().toString();
              int to=pagename.indexOf("@");
                  int from=pagename.lastIndexOf(".");
                  String page=pagename.substring(from+1,
                                                      to-4)+".jsp";
                  int pageno=1;
                  if (allowSorting==null)
                              allowSorting="false";
                  if (allowPaging==null)
                              allowPaging="false";
                  if (recordsPerPage==null)
                              recordsPerPage="10";
                  sortColumnName=getRequestParameter("
                                                    columnname");
                pageno=getRequestParameter("page"));

/* Loading JDBC Driver and constructing Connection object
depending on DBMS */
            if (backEnd==”MS-Access”)
            {
                        loadDriver(“MS-Access”);
                        connectTo(“MS-Access”);
            }

            if (backEnd==”MySQL”)
            {
                        loadDriver(“MySQL”);
                        connectTo(“MySQL”);

            }

           if (backEnd==”Oracle”)
            {
                        loadDriver(“Oracle”);
                        connectTo(“Oracle”);
            }
/* Construct Table Header depending on attributes specified in
Tag */

            query="SELECT * FROM " + tableName;
          executeQuery(query);
            if (sortColumnName != null)
                {
                        query += " ORDER BY ";
                        query += sortColumnName;
                }

            if (allowPaging.equals("true"))
                {
                        if (backEnd.equals("MS-Access"))
                        {
                        if (pageno==1)
                        query="SELECT TOP " + recordsPerPage +
                                    " * FROM " + tableName;
       else
                  query="SELECT TOP " +
recordsPerPage + " * FROM " + tableName + "
WHERE " + colname  + " NOT IN (select  TOP "
 + recordsPerPage * (pageno-1)) + " " + colname
 + " FROM " + tableName +")";
       if (sortColumnName != null)
      {
          query += " ORDER BY ";
          query += sortColumnName;
      }
        }
       else
         {
       query="SELECT * FROM " + tableName;
       if (sortColumnName != null)
            {
                  query += " ORDER BY ";
                  query += sortColumnName;
            }
                   String line= " LIMIT " +
       recordsPerPage * (pageno-1) + "," +
       recordsPerPage;
       query+=line;
         }
}
else
{
       query="SELECT * FROM " + tableName;
       if (sortColumnName != null)
            {
                  query += " ORDER BY ";
                  query += sortColumnName;
            }
}

Iterator it=columns.iterator();
while (it.hasNext())
{
                Column c = (Column)it.next();
                  columnName=c.getName();
                  type=c.getType();
                  url=c.getUrl();
                  hyperText=c.getHyperText();

                  if (allowSorting.equals("true"))
                   {
                              data+="<a href='";
                              data+=page;
                              data+="?columnName=";
                              data+=columnName;
                              if (page != null)
                               {
                                          data+="&page=";
                                          data+=pageno;
                               }
                               data+="'>";
                  data+=columnName;
                  data+="</a>";

                   }
```

```
                        else
                    data+=columnName;
              }

             it=columns.iterator();
           while (it.hasNext())
                        {
                        Column c = (Column)it.next();
                        name=c.getName();
                        type=c.getType();
                        url=c.getUrl();
                         query="SELECT " + name + "
         FROM " + tableName + " WHERE 1 = 2 ";
                              executeQuery(query);
                        }

           while (nextRecord())
           {
                it=columns.iterator();
                while (it.hasNext())
                {
                        Column c = (Column)it.next();
                        name=c.getName();
                        type=c.getType();
                        url=c.getUrl();

                        hyperText=c.getHyperText();
                              if (type.equals("regular"))
                              {
                                          data+=columnValue;
                              }
                            else if (type.equals("checkbox"))
                             {
                              if (columnValue.equals("true"))
                               data+="<input type='checkbox'
                                             checked='true'>";
                                  else
                                     data+="<input
                                           type='checkbox'>";
                                          }
                 else if (type.equals("image"))
                 {
                          data+="<image src='";
                          data+=columnValue;
                          data+="' height=50
                          width=50></img>";
                   }
                else if (type.equals("hyperlink"))
                {
                          data+="<a href='";
                          data+=columnValue;
                          data+="' download>";
                          data+=hyperText;
                          data+="</a>";
                 }
               }
       }
             print(data);
 }
}
```

V. Results and Discussions

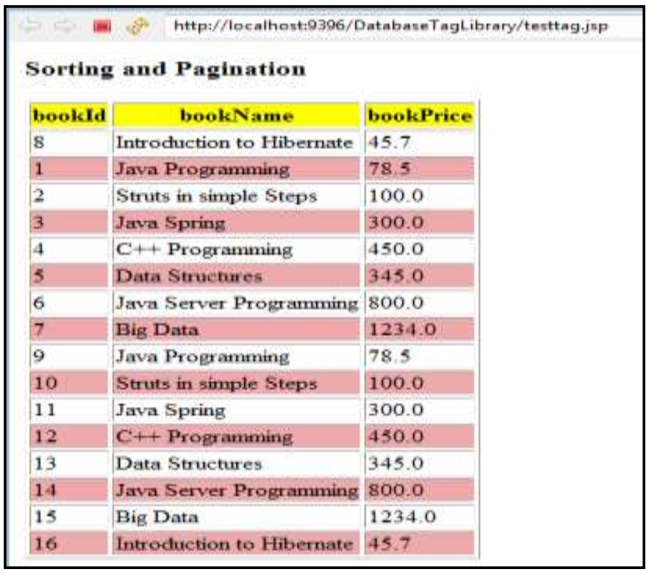
The algorithm presented above is implemented in Java Server Pages using Eclipse editor. The tag is tested for different database management systems. Different test cases are presented here.

Test Case 1 : Without Sorting, Pagination and Filtering Enabled for MySQL Database.

JSP code snippet in which sorting, pagination and filtering are disabled is shown below:

```
<%@ taglib uri="/WEB-INF/lib/customtag.tld"
prefix="Database" %>
<%
  String fid=request.getParameter("fid");
  if (fid == null || fid == "")
      fid="1";
%>
<html>
  <head>
    <title>Custom Tag for Sorting, Pagination and
Filtering</title>
  </head>
  <body>
    <h3>Sorting and Pagination</h3>
    <Database:DisplayTable databaseName="library"
backEnd="MySQL"
            userName="root" password="mca"
tableName="book" columnNames="all"/>
    </body>
</html>
```

GUI generated by the tag is shown in Figure 3.



http://localhost:9396/DatabaseTagLibrary/testtag.jsp

Sorting and Pagination

bookId	bookName	bookPrice
8	Introduction to Hibernate	45.7
1	Java Programming	78.5
2	Struts in simple Steps	100.0
3	Java Spring	300.0
4	C++ Programming	450.0
5	Data Structures	345.0
6	Java Server Programming	800.0
7	Big Data	1234.0
9	Java Programming	78.5
10	Struts in simple Steps	100.0
11	Java Spring	300.0
12	C++ Programming	450.0
13	Data Structures	345.0
14	Java Server Programming	800.0
15	Big Data	1234.0
16	Introduction to Hibernate	45.7

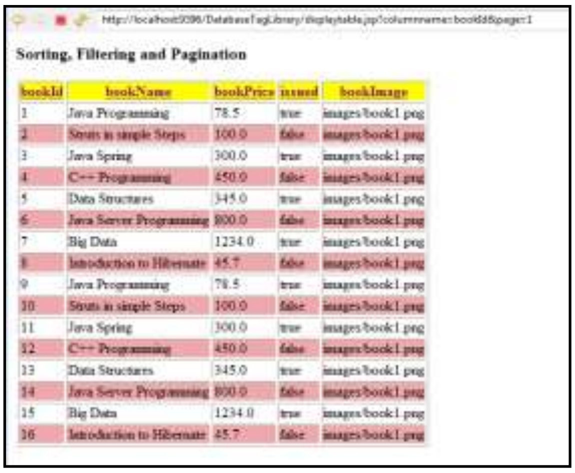
Figure 3. GUI Generated by the Custom Tag on Disabling Sorting, Pagination and Filtering

Test Case 2 : Sorting Enabled – allowSorting attribute of DisplayTableTag tag is set to true for MS-Access Database.

JSP code snippet in which sorting enabled is shown below:

```
<%@ taglib uri="/WEB-INF/lib/customtag.tld" prefix =
"Database" %>
<html>
  <head>
    <title>Custom Tag for Enabling Sorting</title>
  </head>
  <body>
    <h3>Sorting, Filtering and Pagination</h3>
    <Database:DisplayTable dsnName="clibrary"
          tableName="book" columnNames="all"
                     allowSorting="true" />
    </body>
</html>
```

GUI generated by the tag is shown in Figure 4.


http://localhost:9396/DatabaseTagLibrary/displaytable.jsp?columnname=bookId&page=1

Sorting, Filtering and Pagination

bookId	bookName	bookPrice	issued	bookImage
1	Java Programming	78.5	true	images/book1.png
2	Struts in simple Steps	100.0	false	images/book1.png
3	Java Spring	300.0	true	images/book1.png
4	C++ Programming	450.0	false	images/book1.png
5	Data Structures	345.0	true	images/book1.png
6	Java Server Programming	800.0	false	images/book1.png
7	Big Data	1234.0	true	images/book1.png
8	Introduction to Hibernate	45.7	false	images/book1.png
9	Java Programming	78.5	true	images/book1.png
10	Struts in simple Steps	100.0	false	images/book1.png
11	Java Spring	300.0	true	images/book1.png
12	C++ Programming	450.0	false	images/book1.png
13	Data Structures	345.0	true	images/book1.png
14	Java Server Programming	800.0	false	images/book1.png
15	Big Data	1234.0	true	images/book1.png
16	Introduction to Hibernate	45.7	false	images/book1.png

Figure 4. GUI Generated by the Custom Tag with Sorting Enabled

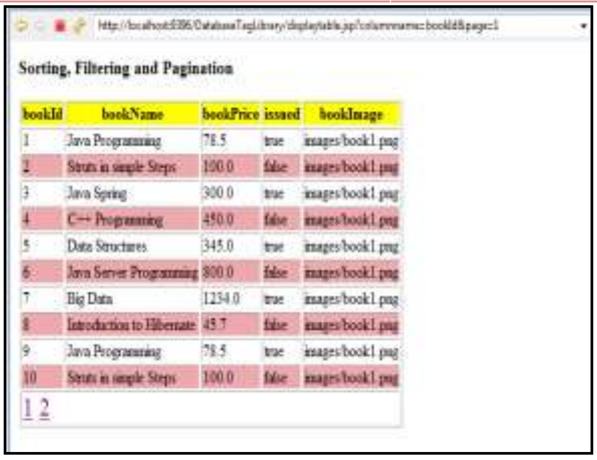
When allowSorting attribute of the tag is set to true the table headers rendered by the tag handler class become hyperlinks. On clicking the column's hyperlink, the data is sorted in ascending order.

Test Case 3 : Pagination Enabled - allowPaging attribute of DisplayTableTag tag is set to true for Oracle 10g Database.

JSP code snippet in which pagination enabled is shown below:

```
<%@ taglib uri="/WEB-INF/lib/customtag.tld"
prefix="Database" %>
<html>
  <head>
    <title>Custom Tags for Database Operations</title>
  </head>
  <body>
    <h3>Sorting, Filtering and Pagination</h3>
    <Database:DisplayTable backEnd="Oracle" userName
="system" password="siber" ipAddress="192.168.30.94"
tableName="book" columnNames="all" allowPaging
="true" />
    </body>
</html>
```

GUI generated by the tag is shown in Figure 5.


http://localhost:9396/DatabaseTagLibrary/displaytable.jsp?columnname=bookId&page=1

Sorting, Filtering and Pagination

bookId	bookName	bookPrice	issued	bookImage
1	Java Programming	78.5	true	images/book1.png
2	Struts in simple Steps	100.0	false	images/book1.png
3	Java Spring	300.0	true	images/book1.png
4	C++ Programming	450.0	false	images/book1.png
5	Data Structures	345.0	true	images/book1.png
6	Java Server Programming	800.0	false	images/book1.png
7	Big Data	1234.0	true	images/book1.png
8	Introduction to Hibernate	45.7	false	images/book1.png
9	Java Programming	78.5	true	images/book1.png
10	Struts in simple Steps	100.0	false	images/book1.png

1 2

Figure 5. GUI Generated by the Custom Tag with Pagination Enabled

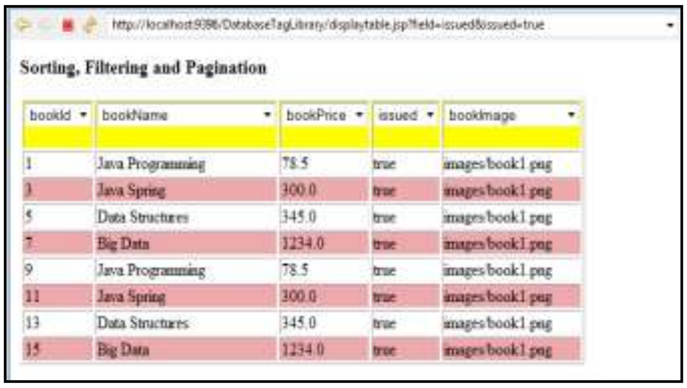
When allowPaging of the attribute is set to true, the first page displays recordsPerPage (default value is 10) no. of records. The user can navigate to any other page by clicking the appropriate page number rendered as a hyperlink in the footer.

Test Case 4 : Filtering Enabled - allowFiltering attribute of DisplayTableTag tag is set to true

JSP code snippet in which pagination enabled is shown below:

```
<%@ taglib uri="/WEB-INF/lib/customtag.tld"
prefix="Database" %>
<html>
  <head>
    <title>Custom Tags for Database Operations</title>
  </head>
  <body>
    <h3>Sorting, Filtering and Pagination</h3>
    <Database:DisplayTable dsnName="clibrary" tableName
="book" columnNames="all" allowFiltering="true" />
    </body>
</html>
```

GUI generated by the tag is shown in Figure 6.


http://localhost:9396/DatabaseTagLibrary/displaytable.jsp?field=issued&issued=true

Sorting, Filtering and Pagination

bookId ▾	bookName ▾	bookPrice ▾	issued ▾	bookImage ▾
1	Java Programming	78.5	true	images/book1.png
3	Java Spring	300.0	true	images/book1.png
5	Data Structures	345.0	true	images/book1.png
7	Big Data	1234.0	true	images/book1.png
9	Java Programming	78.5	true	images/book1.png
11	Java Spring	300.0	true	images/book1.png
13	Data Structures	345.0	true	images/book1.png
15	Big Data	1234.0	true	images/book1.png

Figure 6. GUI Generated by the Custom Tag with Filtering Enabled

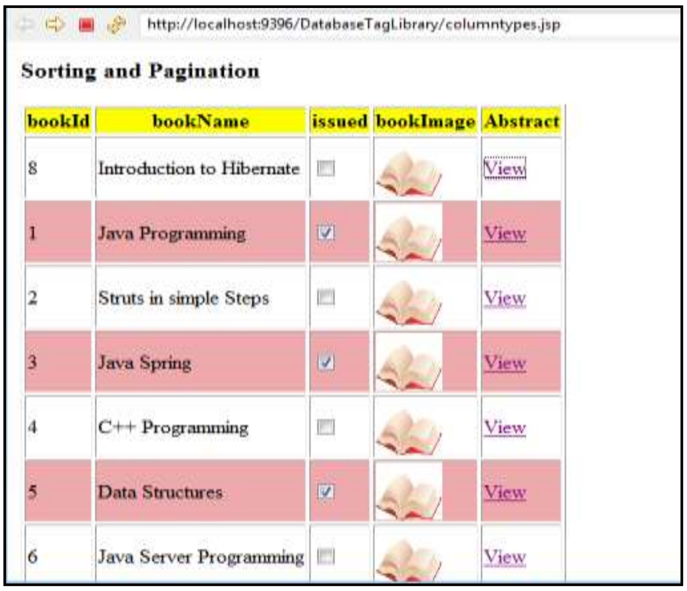
When allowFiltering attribute of the tag is set to true, the drop-down lists appear in the place of column headers which contain unique values in that particular column. The user can select the required value for filtering in the records containing that value.

Test Case 5 : Displaying Columns in Different Formats.

JSP code snippet in which pagination enabled is shown below:

```
<%@taglib uri="/WEB-INF/lib/customtag.tld"
prefix="Database"%>
<html>
<head>
<title>Custom Tags for Database Operations</title>
</head>
<body>
<h3>Sorting and Pagination</h3>
<Database:DisplayTable dsnName="clibrary"
tableName="book">
<Database:Column name="bookId" type="regular"/>
<Database:Column name="bookName" type="regular"/>
<Database:Column name="issued" type="checkbox"/>
<Database:Column name="bookImage" type="image"/>
<Database:Column name="Abstract" type="hyperlink"
hyperText="View"/>
</Database:DisplayTable>
</body>
</html>
```

GUI generated by the tag is shown in Figure 7.


http://localhost:9396/DatabaseTagLibrary/columntypes.jsp

Sorting and Pagination

bookId	bookName	issued	bookImage	Abstract
8	Introduction to Hibernate	☐		View
1	Java Programming	☑		View
2	Struts in simple Steps	☐		View
3	Java Spring	☑		View
4	C++ Programming	☐		View
5	Data Structures	☑		View
6	Java Server Programming	☐		View

Figure 7. GUI Generated by the Custom Tag Set for Displaying Columns in Different Format

As seen from Figure 7. bookId and bookName columns are displayed in regular format whereas issued, bookImage and Abstract columns are displayed in checkbox, image and hyperlink formats.

VI. Conclusion and Scope for Future Work

The current work focuses on creating a custom tag for displaying a table data independent of the database management system in which it is stored in a pageable, sortable and filterable grid view.

In the current work, the entire data from the underlying database is retrieved and the correct subset of records to be displayed is determined at code level, discarding the rest. This is associated with large performance cost since the entire set of records irrespective of the page to be displayed is being returned from the underlying database of which only the desired ones would be displayed. The performance can be greatly enhanced if only the required subset of records which are part of the page to be displayed are returned as it would tremendously reduce a network traffic. In the current work, the custom tag renders a unidirectional sortable grid view where the data is sorted only in ascending order. In future, the tag can be modified to incorporate a bi-directional sortable grid view in which the data can be sorted either in ascending order or descending order. On clicking the hyperlink for the first time, the data is sorted in ascending order and the data is sorted in descending order if the same hyperlink is clicked again

References

[1] Dr. Poornima G. Naik, JSP Custom Tag Library for Implementing JDBC Functionality, http://www.codeproject.com/Articles/1084607/JSP-Custom-Tag-Library-for-Implementing- JDBC-Funct, 11th March 2016.

[2] Dr. Poornima G. Naik, JSP Custom Tag Library (Version 2) for DML Operations, http://www.codeproject.com/Articles/1085185/JSP-Custom-Tag-Library-Version-for-DML-Operations, 14th March, 2016

[3] Dr. Poornima G. Naik, JSP Custom Tag Library for Table Joins and Master Detail Relationships, http://www.codeproject.com/Articles/1086716/JSP-Custom-Tag-Library-for-Table-Joins-and-Master, 19th March, 2016.

[4] Xiong, Yingyidu. "The design of automatically generating drop-down a menu on JSP." Computer Science and Information Processing (CSIP), 2012 International Conference on. IEEE, 2012.

[5] Gupta, Karan, and Anita Goel. "Requirement Estimation and Design of Tag software in Web Application." International Journal of Information Technology and Web Engineering (IJITWE) 9.2 (2014): 1-19.

[6] Cuthbertson, Daniel J., et al. "Accurate mass–time tag library for LC/MS based metabolite profiling of medicinal plants." Phytochemistry 91 (2013): 187-197.

[7] Liu, Man, Xiaohui Wu, and Qingshun Quinn Li. "DNA/RNA Hybrid Primer Mediated Poly (A) Tag Library Construction for Illumina Sequencing."Polyadenylation in Plants: Methods and Protocols (2015): 175-184.

[8] Kamal, Arif. "Tag Based Audiovisual Content Indexing.", MASTER'S THESIS, Master of Science, Computer Science and Engineering,Luleå University of Technology, Department of Computer science, Electrical and Space engineering, 2016

[9] Xu, Ming, et al. "Research on the Method of Code Migration from JSP to ASP. NETMing." Advanced materials research. Vol. 756. 2013.

[10] Nam Ky Giang, Michael Blackstock, Rodger Lea, Victor C.M. Leung , Developing IoT Applications in the Fog: a Distributed Dataflow Approach. Procs. of the Internet of Things (IOT), 2015 International Conference on the, Seoul, Korea, Oct 26-28, 2015

[11] Dr. Poornima G. Naik and Dr. K.S.Oza, JSP Custom Tag Library for In-Place Editing in Disconnected Architecture - A Case Study, International Journal on Recent Trends in Computing and Communication, vol. 4, no. 4, pp. 319-326, April 2016. Barni M., Bartolini F., Piva A., Multichannel watermarking of color images, IEEE Transaction on Circuits and Systems of Video Technology 12(3) (2002) 142-156.

[12] Dr. Poornima G. Naik, Mr. Girish R. Naik, JSP Custom Tag for Displaying Master- Detail Relationship in a Hierarchical Grid Control – A Case Study, International Journal of Engineering Applied Sciences and Technology, vol. 1, no 8, pp. 65-71, June 2016.