International Journal of Engineering Applied Sciences and Technology, 2016
Vol. 1, Issue 8, ISSN No. 2455-2143, Pages 65-71
Published Online June - July 2016 in IJEAST (http://www.ijeast.com)

JSP CUSTOM TAG FOR DISPLAYING MASTER-DETAIL RELATIONSHIP IN A HIERARCHICAL GRID CONTROL – A CASE STUDY

Dr.Poornima G. Naik
Department of Computer Studies
Chh. Shahu Institute of Business Education & Research
Kolhapur, Maharastra, India

Mr. Girish R. Naik
Production Department
KIT' College of Engineering
Kolhapur, Maharastra, India

Abstract— **Applications of all type and size incorporate some sort of database functionality and reveal implicit relationship between different parts of data. This type of relationship can very well be modeled using master-detail relationship. All modern GUI-based languages support rich controls for visualizing master-detail relationships. One such popular control is a hierarchical grid control which expands and collapses the detailed records based on the selection of the master record. One of the prime benefits offered by the custom controls is that they are reusable within the platform. However, such controls suffer from the severe limitation that they are resource intensive and are not portable. The best of both the worlds can be realized by implementing the required functionality as a part of custom tag library. Reusability can effectively be blended with reusability in JSP custom tag library. Most of the popular frameworks rely on their own custom tag libraries for GUI design, code template, validation etc. to name a few. In order to cater this general demand, in this paper, the authors have presented JSP custom tag library for displaying master-detail relationship in a hierarchical data grid control. In contrast to other custom controls, this particular control is drawn which renders it light-weight and amenable to be incorporated in web applications. The tag works with disparate back ends and can comfortably be extended to other back ends with only minor changes in the code. The database extensions are properly taken care of by pulling out the relevant schema information from the underlying database management system.**

Keywords— **Attribute Design Pattern, Back End, BodyTagSupport class, Custom Control, Nested Tags, Tag Handler Class, TagLib Directive.**

I. INTRODUCTION

One of the prime benefits offered by object-oriented programming paradigm is reusability of code. Nevertheless there is a fistful of technologies supported by modern languages which result in plethora of boilerplate code required towards exception handling and code management. One such technology supported by Java is JDBC technology. Meanwhile many alternatives have been proposed for encapsulating such boilerplate code by reducing them to one liners. One such technique is implementation of custom tag library.

JSP technology supports tag extensions where a new mechanism is devised for creating new extensions and embedding them in a JSP page focusing on the tag simplicity rendering them usable by the non-programmers.

The prime steps involved in writing a custom tag are:

- Designing and implementing a tag handler class for the custom tag. This can be achieved by extending either ***javax.servlet.jsp.tagext.TagSupport*** or ***javax.servlet.jsp.tagext.BodyTagSupport*** class.
- Storing the tag specific information in an XML file by generating tag library descriptor document.
- Designing a client JSP page by employing taglib directive.

The inheritance diagram for the BodyTagSupport class is depicted in Fig. 1.

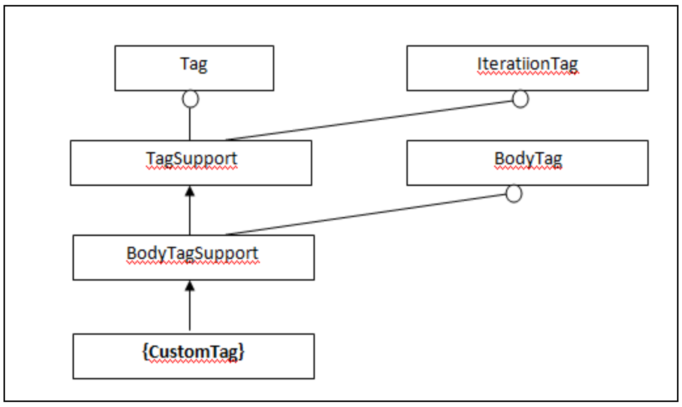


Fig. 1. The inheritance diagram for the BodyTagSupport class

As seen from Figure 1 a custom tag can be created by extending a BodyTagSupport class which in turn extends TagSupport class and implements BodyTag interface. TagSupport class further implements either Tag or IternationTag interface.

JSP action tags facilitate rapid application development by concealing a huge amount of boilerplate code behind a set of custom tags. One of the authors is involved in the development of custom tag library for displaying table data in a database independent manner, performing various DML operations, performing table joins etc. [1,3].

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 thorough 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

A. Design Pattern for Tag Attribute –

The design pattern for a typical tag attribute consists of a pair or accessor and mutator methods as shown below:

```
T getN();
void setN(T)
```

where N is the name of the tag attribute and T is its type.

B. Tag Library Descriptor Document –

A tag library descriptor file is a simple XML file with the extension .tld embedding a set of custom tags encapsulating a tag related information. 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.

C. Life Cycle of a Tag Handler Class –

The life cycle of a tag is depicted in Figure 2. As shown in Fig. 2. setParent() method is invoked only for nested tags. Mutator methods are invoked for every attribute specified in the tag. doStartTag() method is invoked when the start tag is encountered which returns one of the constants SKIP_BODY or EVAL_BODY_INCLUDE. In the former case, doEndTag() method is invoked while in the latter case doAfterBody() is invoked which is iteratively invoked till the SKIP_BODY is returned after which a concluding life cycle method doEndTag() is invoked.

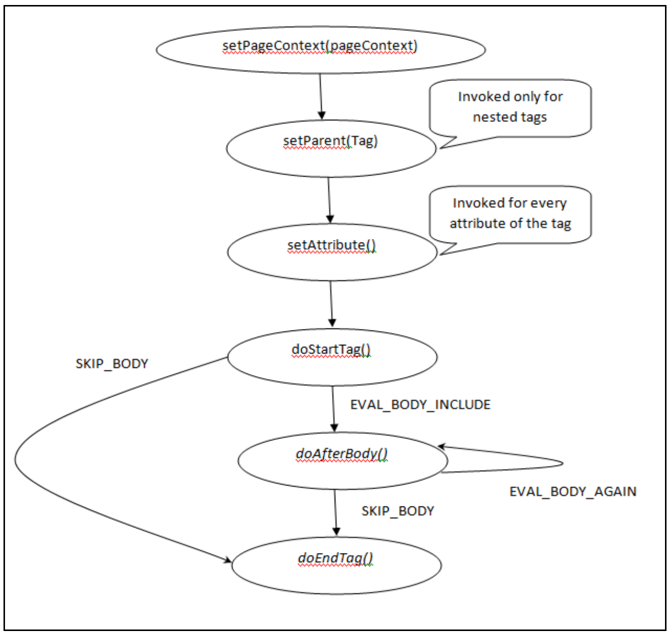


Fig. 2. Life Cycle of Custom Tag

is i JSP tags can be nested one inside another. In such a case, an inner tag can access the attributes of an outer tag by retrieving a reference to the outer tag handler class and then invoking the corresponding accessor methods of tag handler class as shown in the following code snippet.

```
ParentTag p = (ParentTag)
findAncestorWithClass(this, ParentTag.class);
T p.getN();
```

where ParentTag refers to the outer tag in which the current tag is nested and N is the property of its tag handler class of type T corresponding to the attribute of the tag.
Another requirement is that the doStartTag() method of parent tag's tag handler class should return a constant ***EVAL_BODY_INCLUDE***, which evaluates body into existing out stream.

D. Communication Between a Tag Handler Class and a Client JSP Page –

The expand and collapse buttons on the hierarchical grid control placed on a client JSP page cause a post back to the same JSP page. During the self post back, the hierarchical grid control is updated each time to show/hide detail records as the case may be. Since the control is rendered by the corresponding tag handler class, the communication between client JSP page and the corresponding tag handler class becomes inevitable where the client JSP page passes on the end user action information to the tag hander class specifying the type of operation the user desires to perform at runtime. The following code snippet is employed for extracting the name of the client JSP page from within the tag handler class.

```
String pagename=this.pageContext.getPage().toString();
int to=pagename.indexOf("@");
int from=pagename.lastIndexOf(".");
String page=pagename.substring(from+1, to-4)+".jsp";
```

Now, the tag handler class can use this information for rendering the control properly.
For rendering the tag generic, the attributes applicable to the various database management systems are identified and are added to the parent tag definition in the tag library descriptor file [11].

E. Control Folder Structure–

The position of the different project components in an Eclipse project folder with the name hierarchical grid is depicted in Fig. 3.

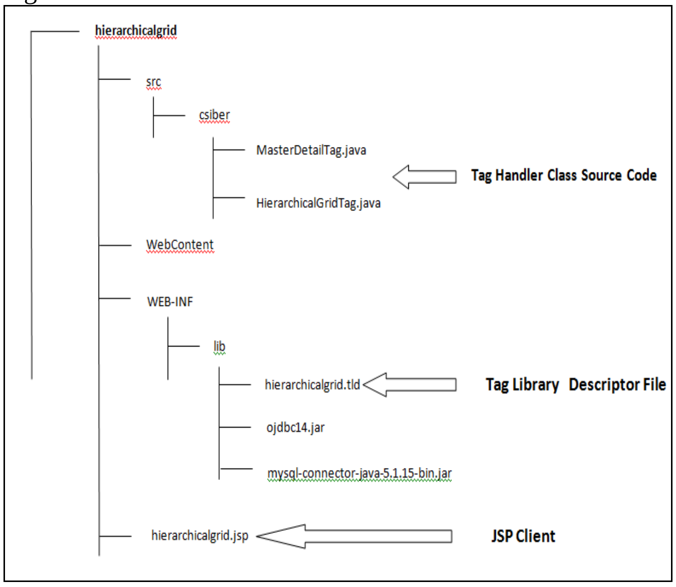


Fig. 3. Control Folder Structure

F. Class Diagram –

The structure of the different classes employed in control design is shown in Fig. 4.

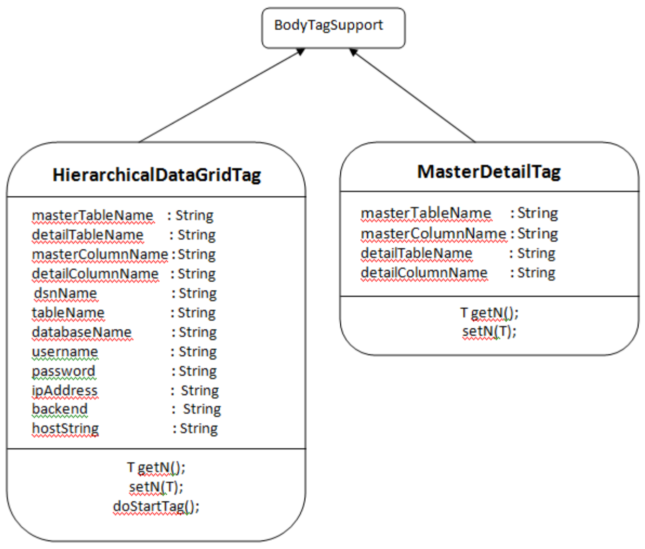


Fig. 4. Structure of Tag Handler Classes

G. Interaction Between Project Components –

Figure 5 depicts the interaction between project components. As seen from Figure 5, JSP runtime invokes JSP client which embeds custom tag employing taglib directive and the tag is processed by the JSP runtime by invoking the corresponding tag handler class and accessing tag library descriptor file which includes tag description. The tag handler class employs JDBC drivers for interfacing with back end database management systems.

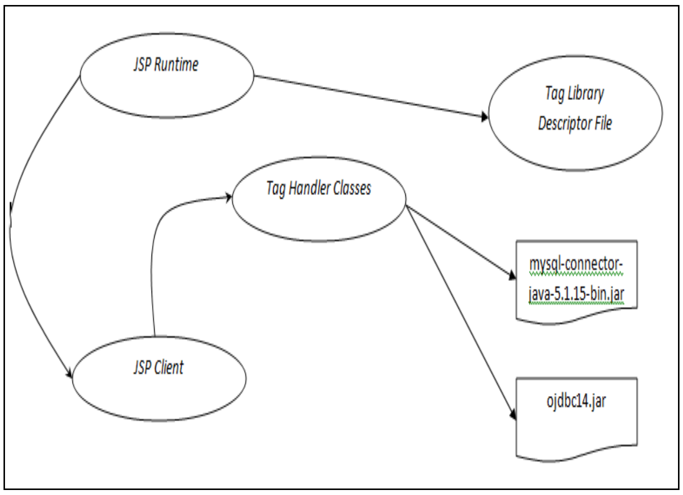

JSP Runtime
Tag Library Descriptor File
Tag Handler Classes
mysql-connector-java-5.1.15-bin.jar
JSP Client
ojdbc14.jar

Fig. 5. Interaction Between Project Components

IV. Implementation of a Custom Tag

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

A. Structure of Column Tag –

Structure of MasterDetail tag

```
<tag>
   <name>MasterDetail</name>
   <tag-class>csiber.MasterDetailTag</tag-class>
   <body-content>JSP</body-content>
    <attribute>
     <name>masterTableName</name>
   </attribute>
   <attribute>
     <name>masterColumnName</name>
   </attribute>
   <attribute>
     <name>masterColumnValue</name>
     <rtexprvalue>true</rtexprvalue>
   </attribute>
<attribute>
     <name>detailTableName</name>
   </attribute>
   <attribute>
     <name>detailColumnName</name>
   </attribute>
  </tag>
```

Structure of HierarchicalDataGrid tag

```
<tag>
   <name>HierarchicalDataGrid</name>
   <tag-class>csiber.HierqarchicalDataGridTag</tag-class>
   <body-content>empty</body-content>
   <attribute>
     <name>dsnName</name>
   </attribute>
   <attribute>
     <name>dsnName</name>
   </attribute>
      <attribute>
     <name>backEnd</name>
   </attribute>
   <attribute>
     <name>databaseName</name>
   </attribute>
   <attribute>
     <name>userName</name>
   </attribute>
   <attribute>
     <name>password</name>
   </attribute>
  </tag>
```

B. 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.
*/

```
/* Invoked wheh start tag is rendered */
function doStartTag()
{
        Read backEnd;
        if (backEnd==null)
        {
                backEnd=”MS-Access”;
        }
        /* Load appropriate database driver and construct
database connection */

        if (backEnd==”MS-Access”)
        {
                loadDriver(“MS-Access”);
                connectTo(“MS-Access”);
        }

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

if (backEnd==”Oracle”)
        {
                loadDriver(“Oracle”);
                connectTo(“Oracle”);
        }

        /* Extract the name of the page for self postback */
        String page=getPageName();
         /* Extract Query String Parameter oper */
        String oper=getQueryString(“oper”);
        if (startsWith(oper, “-“))
                {

        query=constructQuery(detailTableName);
                                executeQuery(query);
                                metadata=getResultSetMetadata();
                                if (!startsWith(oper,”-“))
                                row=substring(oper,1);
                                if (!startsWith(oper,”-“))
                                        oper=”+”;
                                else
                                        oper=”-“;

        query=constructQuery(masterTableName);
                                executeQuery(query);
                                        if (row==” ”)
                {
query=constructQuery(detailTableName);
                                executeQuery(query);
                                displayDetailRecords();
                }
              }
           }
       }
}
```

V. Results and Discussions

The algorithm presented in Section 4 is implemented in Java using Eclipse editor. The tag is tested for three different database management systems MS-Access, MySQL, and Oracle 10g.
The code snippet for accessing the tag in each case is shown below:

A. JSP Client for MS-Access –

In order to display master-detail relation in a hierarchical grid control for MS-Access database create a 32-bit system DSN on windows platform. The JSP code for achieving this is shown below:

```
<%@ taglib uri="/WEB-INF/lib/hierarchicalgrid.tld"
prefix="Database" %>
<html>
```

```
<head>
<title>Masterr-Detail Relationship in Hierarchical Grid
Control</title>
</head>
<body>
<h3>Hierarchical Grid control</h3>
<Database:MasterDetail masterTableName="student"
masterColumnName="RollNo" detailTableName="marks"
detailColumnName="RollNo" >
<Database:HierarchicalDataGrid  dsnName="exam"/>
</Database:MasterDetail>
</body>
</html>
```

B. **JSP Client for MySQL –**

For interfacing with MySQL database management system appropriate Type-IV JDBC driver needs to be downloaded and be placed in WEB-INF\lib folder under a project folder.

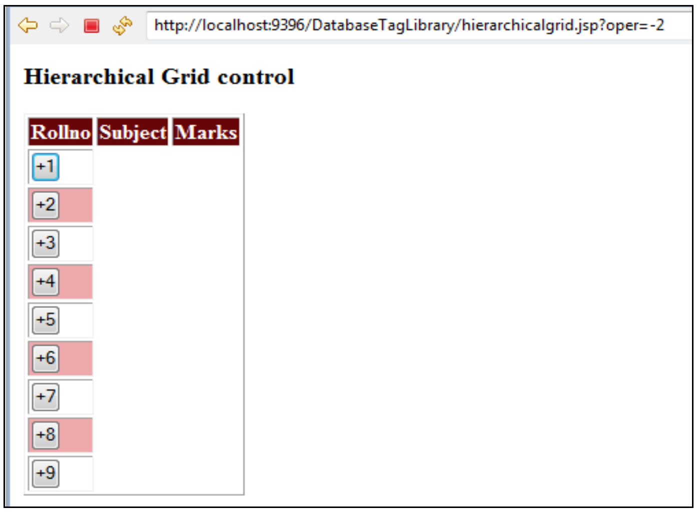
```
<%@        taglib        uri="/WEB-INF/lib/hierarchicalgrid.tld"
prefix="Database" %>
<html>
  <head>
    <title>Masterr-Detail Relationship in Hierarchical Grid
Control</title>
  </head>
  <body>
    <h3>Hierarchical Grid control</h3>
    <Database:MasterDetail masterTableName="student"
masterColumnName="RollNo" detailTableName="marks"
detailColumnName="RollNo" >
        <Database: HierarchicalDataGrid
databaseName="exam"  backEnd="MySQL"
        userName="root" password="mca" />
    </Database:MasterDetail>
</body>
</html>
```

C. **JSP Client for Oracle –**

In order to interface with Oracle 10g database management system oracle thin driver of Type-IV JDBC driver, ojdbc.jar needs to be downloaded and be placed in WEB-INF\lib folder under a project folder.

```
<%@        taglib        uri="/WEB-INF/lib/hierarchicalgrid.tld"
prefix="Database" %>
<html>
  <head>
    <title>Masterr-Detail Relationship in Hierarchical Grid
Control</title>
  </head>
  <body>
    <h3>Hierarchical Grid control</h3>
    <Database:MasterDetail masterTableName="student"
masterColumnName="RollNo" detailTableName="marks"
detailColumnName="RollNo" >
        <Database: HierarchicalDataGrid backEnd="Oracle"
              userName="system"  password="siber"
            ipAddress="192.168.30.94"  hostString="orcl"/>
      </Database:MasterDetail>
<body>
</html>
```

The Graphical User Interface (GUI) which is dynamically generated by the custom tag is shown in Fig. 6(a) and 6(b). Fig. 6(a) depicts the state of hierarchical grid control in collapse state and Fig. 6(b) depicts the state of hierarchical grid control in an expanded state on selection of a particular master table record.

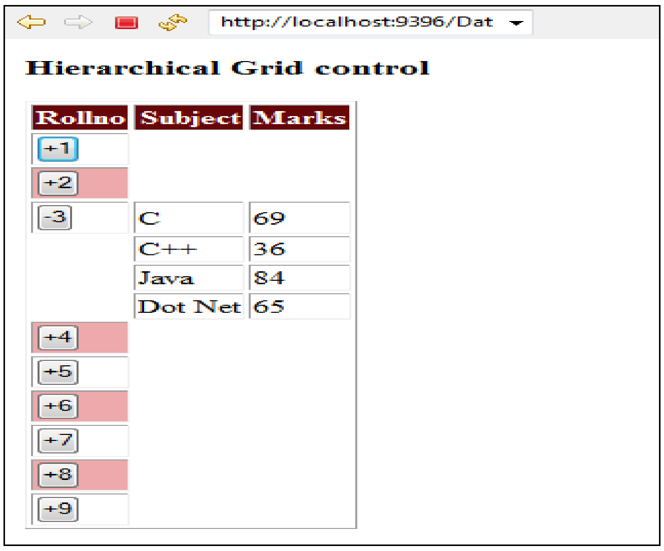


http://localhost:9396/DatabaseTagLibrary/hierarchicalgrid.jsp?oper=-2

Hierarchical Grid control

Rollno	Subject	Marks
+1		
+2		
+3		
+4		
+5		
+6		
+7		
+8		
+9		

Fig. 6(a) Hierarchical Grid Control in Collapse State



http://localhost:9396/Dat

Hierarchical Grid control

Rollno	Subject	Marks
+1		
+2		
-3	C	69
	C++	36
	Java	84
	Dot Net	65
+4		
+5		
+6		
+7		
+8		
+9		

Fig. 6(b) Hierarchical Grid Control in Expanded State

VI. CONCLUSION AND SCOPE FOR FUTURE WORK

JSP custom tags play a prominent role in code reduction and code reusability and make significant contribution to rapid web application development. As the tag is implemented in

Java, the tag automatically reaps the benefits offered by Java language such as portability, security and robustness. In the current work the authors have designed and implemented a custom tag for modeling the master-detail relationship in a hierarchical grid control. The algorithm is devised for control design and control folder structure and interaction between various project components is presented. The current work can be extended further to incorporate multiple levels of hierarchy in a single control for modeling deeply nested relations.

VII. REFERENCE

[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.