International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 5 Issue: 3 261 – 268

Struts, Hibernate and Spring Integration – A Case Study

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

Mr. Girish R. Naik
Associate Professor
Production Department
KIT's College of Engineering
Kolhapur, India

Abstract— Over the last few decades software development has undergone tremendous radical changes in order to enhance user experience, user friendliness and to widen its scope over large geographical area. The key actors in this arena are two leading IT companies Microsoft and Sun Microsystems, now taken owner by Oracle Inc. who compete on a continual basis for rendering rapid professional software design and development process at the same time incorporating more functionality and focusing strongly towards software maintenance issues. Due to this intense competition, the ultimate beneficiary is a software developer who is tremendously benefited at large. In this review paper, the researcher aims at consolidating the technological advancements that have brought a revolutionary change in corporate software development over last few decades. The main substance of the paper is technological advancements facilitating J2EE application development viz., struts framework, hibernate and spring framework which operate in different layers of scalable N-tier architecture. Each technology has its own merits and de-merits. The researcher attempts to aggregate the benefits offered by the trio in a single J2EE application thereby bringing in best of three worlds to a single application. The application is boosted with powerful struts tag library, persistent layer provided by hibernate and dependency injection or Inversion of Control (IoC) user the same roof. A case study is presented to demonstrate the integration of three diverse technologies, struts, hibernate and spring. JBOSS application server is employed as a container for J2EE components.

Keywords-*Model-View-Controller Architecture, Loose Coupling, Object-Relation Mapping, Persistent Layer, Plain Old Java Objects, Tight Coupling,*

*****

I. INTRODUCTION

Each of Struts2, Spring and Hibernate is a diverse open source technology targeting different objectives. Each technology has its own advantages. Apache Struts 2 framework is an open-source web application framework for developing Java EE web applications adopting a model–view–controller (MVC) architecture which extends a Java Servlet API. Struts 2 offers configurable MVC components. POJO (Plain Old Java Objects) based actions, rich tab library support make Struts2 a favourite framework among developers for building J2EE applications in MVC architecture. Hibernate is a high-performance Object/Relational persistence and query service which is licensed under the open source GNU Lesser General Public License (LGPL) and is free to download. Hibernate is commonly adopted for developing a persistent layer of a J2EE application. The Spring Framework consists of features organized into about 20 modules. These modules are grouped into Core Container, Data Access/Integration, Web, AOP (Aspect Oriented Programming), Instrumentation, Messaging, and Test. The major benefits offered by Spring framework deal with separation of dependencies and their injection into the application at runtime. Aspect –oriented programming (AOP) on the other hand separate the cross-cutting concerns which span across multiple points in multiple modules from the main business logic and weaving them in to the main application either at compile time, class load time or runtime.

Spring is a popular web framework which supports an easy integration with external frameworks as such does not impose any restriction with framework selection, instead works as a glue for holding together disparate frameworks in a single application. Spring is much more than an MVC framework. It offers many benefits which are lacking in Struts framework. The current work focuses on bringing all the three diverse technologies to a common platform so that a single application can reap the benefits of all the three. Hibernate is employed for designing a persistent layer on the top of MVC architecture of Struts framework where the dependencies are injected dynamically at runtime using IoC (Inversion of Control) of Spring Framework

II. LITERATURE REVIEW

The journey of software development process from few decades back to till date is not abrupt. The development process has witnessed dramatic changes over a period. Irrespective of the type of the key player involved in technology transfer the common goal was to target multiple platforms, multiple devices, support of large user base and internationalization. The least common denominator underlying all these changes include:

- Separation of code from its presentation
- Separation of business logic from presentation logic

 ISSN: 2321-8169
261 – 268

- Separating out software concerns
- Addresssing State managements techniques
- Addressing security issues
- Incorporating efficiency improvement techniques
- Addressing transaction control and concurrency control mechanisms
- Hiding lot of repeated code behind a set of libraries

Java technology started as a single large player in a software development solving platform dependency issues, security issues, eliminating lacunae present in the language, adding web flavour to software development through applets early in 1980s. 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 [1-4]. The concept of Annotations in Java with an emphasis on various in-built annotations in Java and the annotations that are used by other annotations is reported [5]. The reader is introduced to J2EE standard annotations and those employed by Hibernate as a replacement for XML-based mapping document. Steps in designing and using custom annotations are highlighted. A custom annotation design is illustrated with the help of an example for execution of DML commands in a generic way in a database management system independent manner. The documentation of various object-oriented framework and relative comparison between different model of MVC architecture is reported in literature [6, 7]. Implementation of MVC using spring and struts framework was carried out by Gupta et. al.. [8]. Integration of multiple frameworks for an E-commerce system and University system has been addressed [9-11].

III. IMPLEMENTATION DETAILS

A. Application Folder Structure

The trivial partial class diagram of the classes employed in the application is shown in Figure 1.

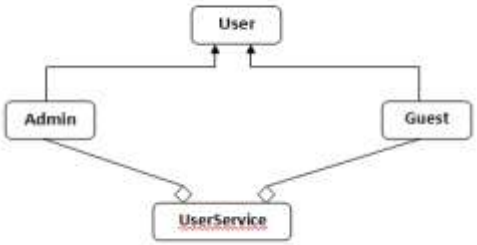

Figure 1. Partial Class Diagram

The folder structure and different components constituting the application is depicted in Figure 2.

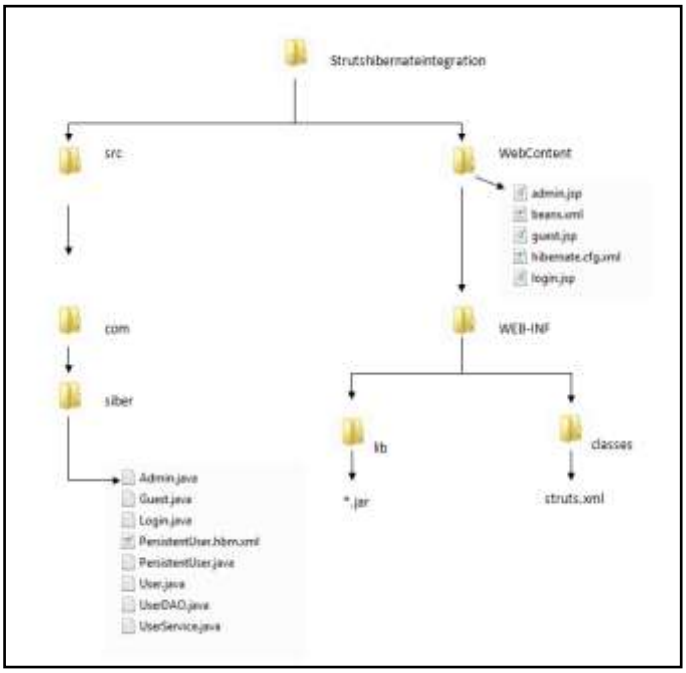

Figure 2. Application Folder Structure

The work flow of the application between different frameworks and between different components of each framework is depicted in Figure 3.

B. Application Work Flow

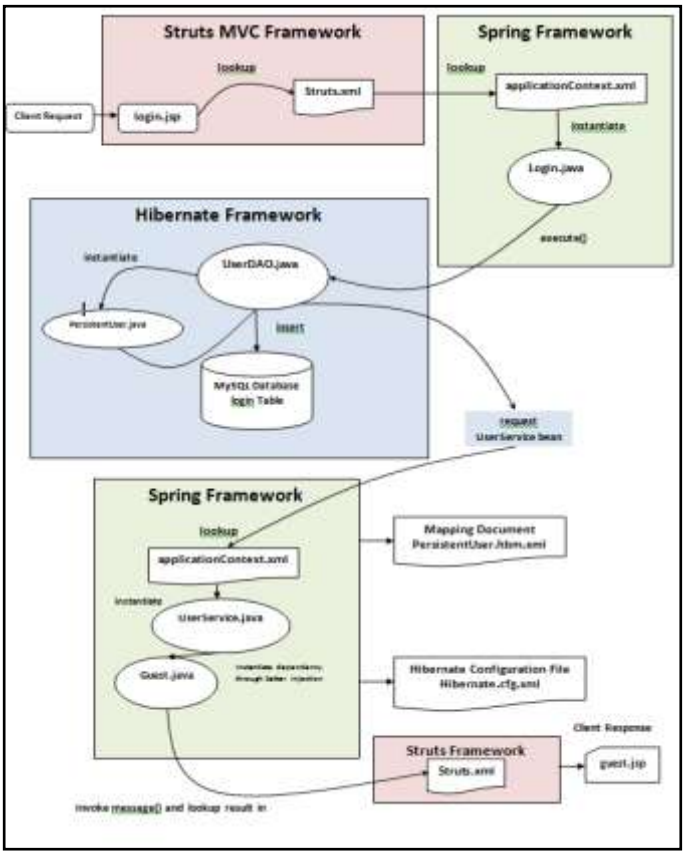

Figure 3. Application Work Flow

 ISSN: 2321-8169
261 – 268

C. Insight into Execution of Application

The following steps are involved in the execution of application.

1. The user requests login.jsp page which contains the line
   <s:form action="user">

2. The struts configuration file struts.xml is looked up for action with the name user and the following line is encountered.

<action name="user" class="loginClass" method="execute">

3. The spring configuration file applicationContext.xml is looked up for the bean with the id loginClass and the following line is encountered.

<bean id="loginClass" class="com.siber.Login"/>

4. Login class is loaded by the class loader and is instantiated.

5. SetterX() methods are invoked for initializing the userName and password with the values entered by the user in login.jsp page.

6. Next, Login class's execute() method is invoked.

7. The first statement in execute( ) method invokes the static method saveUser() of UserDAO class which creates as object of PersistentUser class by invoking parameterized constructor for initializing userName and password and employs hibernate classes for persisting data into a MySQL table user.

8. The following statements in execute() method dynamically instantiate UserSerive class with its dependency Guest stored in User class reference by reading the corresponding information from applicationContext.xml file as shown below:

```
<bean id="guest" class="com.siber.Guest"/>
<bean id="admin" class="com.siber.Admin"/>
<bean id="userService" class="com.siber.UserService">
<property name="user">
<ref bean="guest"/>
</property>
</bean>
```

9. message( ) method of Guest class is invoked which returns the string "guest". The same is returned by the execute() method of UserService class.

10. The struts configuration file strusts.xml is looked up for the result equal to "guest" and the following line is encountered.

<result name="guest">/guest.jsp</result>

11. The view specified in struts.xml file, guest.jsp is displayed next and the response is generated to an end user.

D. Complete Source Code

User.java

```
package com.siber;
public class User {
        String userName;
        public String message()
        {
                return null;
        }

        public String getUserName() {
          return userName;
        }

        public void setUserName(String userName) {
          this.userName = userName;
        }
      }
```

Admin.java

```
package com.siber;

public class Admin extends User{
        public String message()
        {
                return "admin";
        }
}
```

Guest.java

```
package com.siber;

public class Guest extends User{

        public String message()
        {
                return "guest";
        }
}
```

Login.java

```
package com.siber;
import
org.springframework.context.support.ClassPathXmlApplicatio
nContext;

public class Login {
private String userName;
private String password;

public String getUserName() {
        return userName;
}

public void setUserName(String userName) {
        this.userName = userName;
}

public String getPassword() {
```

 ISSN: 2321-8169
261 – 268

```
		return password;
}

public void setPassword(String password) {
		this.password = password;
}

public String execute()
{
		try
		{
		  UserDAO.saveUser(userName, password);
		ClassPathXmlApplicationContext appContext=new
ClassPathXmlApplicationContext( "/WEB-
INF/applicationContext.xml");

		UserService
us=(UserService)appContext.getBean("userService");
		//UserService us=new UserService();
			//us.setUser(new Guest());
		return us.message();
		}
		catch(Exception e)
		{
			System.out.println(e);
			return null;
		}

		//return "guest";
}
}
```

PersistentUser.java

```
package com.siber;
import java.io.Serializable;

public class PersistentUser implements Serializable {
		private int id;
		private String userName ;
		private String password;

		public PersistentUser()
		 {
		 }

		public PersistentUser(String userName, String
password)
		 {
			this.userName=userName;
			this.password=password;
		 }

		public int getId()
		{
			return id;
		}

		public void setId(int id)
		{
			this.id=id;
		}
		public String getUserName() {
			return userName;
		}

		public void setUserName(String userName) {
			this.userName = userName;
		}

		public String getPassword() {
			return password;
		}

		public void setPassword(String password) {
			this.password = password;
		}

}
```

UserDAO.java

```
package com.siber;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;
public class UserDAO {
		public static void saveUser(String userName, String
password){

			SessionFactory sessionFactory;
			  Session hibSession;



			sessionFactory=new
Configuration().configure().buildSessionFactory();
			hibSession=sessionFactory.openSession();
			  Transaction tx=null;

			try
			  {
			tx=hibSession.beginTransaction();
			PersistentUser u = new
PersistentUser(userName, password);
			hibSession.save(u);
			tx.commit();
			}
			catch(RuntimeException e)
			{
				if (tx != null)
					tx.rollback();
				throw e;
			}
		}
		}
```

UserService.java

```
package com.siber;
```

 ISSN: 2321-8169
261 – 268

```
public class UserService {
        User user;

        public void setUser(User user)
        {
                this.user=user;
        }

        public String message()
        {
                return user.message();
        }
}
```

E. Structure of XML Configuration Files

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee/web-
app_2_5.xsd"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
id="WebApp_ID" version="2.5">
  <listener>
    <listener-class>
org.springframework.web.context.ContextLoaderListener
    </listener-class>
  </listener>
  <filter>
    <filter-name>struts2</filter-name>
    <filter-class>
org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecut
eFilter
    </filter-class>
  </filter>
  <filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
</web-app>
```

struts.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration
2.0//EN"
"http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
  <constant name="struts.devMode" value="true" />
  <package name="helloworld" extends="struts-default">
    <action name="user" class="loginClass"
method="execute">
      <result name="admin">/admin.jsp</result>
      <result name="guest">/guest.jsp</result>
    </action>
  </package>
</struts>
```

applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://www.springframework.org/schema/p"
xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:tx="http://www.springframework.org/schema/tx"
xsi:schemaLocation="http://www.springframework.org/schem
a/beans
http://www.springframework.org/schema/beans/spring-beans-
2.5.xsd
      http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-
2.5.xsd
      http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-
2.5.xsd">
   <bean id="guest" class="com.siber.Guest"/>
  <bean id="admin" class="com.siber.Admin"/>
 <bean id="userService" class="com.siber.UserService">
  <property name="user">
   <ref bean="guest"/>
  </property>
  </bean>
   <bean id="loginClass" class="com.siber.Login"/>
</beans>
```

PersistentUser.hbm.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-
//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
<class name="com.siber.PersistentUser" table="login"
catalog="test">
<id name="id" type="java.lang.Integer">
<column name="id"/>
<generator class="identity"/>
</id>
<property name="userName" type="java.lang.String">
<column name="username" length="50"/>
</property>
<property name="password" type="java.lang.String">
<column name="Password" length="50"/>
</property>
</class>
</hibernate-mapping>
```

hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-
//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://hibernate.sourceforge.net/hibernate-configuration-
3.0.dtd">
<hibernate-configuration>
<session-factory>
```

 ISSN: 2321-8169
261 – 268

```
<property
name="connection.driver_class">com.mysql.jdbc.Driver</pro
perty>
<property
name="connection.url">jdbc:mysql://localhost:3306/test</pro
perty>
<property name="connection.username">root</property>
<property name="connection.password">mca</property>
<property
name="dialect">org.hibernate.dialect.MySQLDialect</propert
y>
<mapping resource="com/siber/PersistentUser.hbm.xml" />
</session-factory>
</hibernate-configuration>
```

F. Client –Side Code

login.jsp

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="s" uri="/struts-tags" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Hello World</title>
</head>
<body>
  <h1>Struts2 - Spring and Hibernate Integration</h1>
  <s:form action="user">
    <s:textfield name="userName" label="User Name"/><br/>
    <s:password name="password" label="Password"/><br/>
    <center><s:submit value="Login"/></center>
  </s:form>
</body>
</html>
```

admin.jsp

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h1><font color=green>Admin Console</font></h1>
<table border width="80%">
<tr>
  <td><a href="">User Maintenance</a></td>
  <td><a href="">Stock Maintenance</a></td>
  <td><a href="">View Reports</a></td>
  <td><a href="">Sign In</a></td>
</tr>
</table>
</body>
</html>
```

guest.jsp

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h1><font color=blue>Welcome Guest </font></h1>
<table border width="80%">
<tr>
  <td><a href="">Browse Products</a></td>
  <td><a href="">Sign Out</a></td>
</tr>
</table>
</body>
</html>
```

IV. RESULTS AND DISCUSSIONS

A. Testing of the Application

The user generates an HTTP request, requesting the login.jsp page. The HTTP response generated by JBOSS application server is depicted in Figure

Figure 4. End User Requesting Login Page

The hibernate framework constituting the persistent layer of the application retrieves the data input by the user and inserts a corresponding record in login table of MySQL database as shown in Figure 5.

```
mysql> use test
Database changed
mysql> select * from login;
+----+----------+----------+
| id | username | password |
+----+----------+----------+
|  1 | mca      | mca      |
|  2 | X        | X        |
|  3 | mba      | mba      |
|  4 | msw      | msw      |
|  5 | admin    | admin    |
+----+----------+----------+
5 rows in set (0.19 sec)
```

Figure 5. Inserting a Row in MySQL Table

Case 1 :Tight Coupling

Test Run 1 : Implement the execute() method of Login class as shown below:

UserService us=new UserService();
us.setUser(new Guest());

Execute the application. The output generated is shown in Figure 6..

Figure 6. Tight Coupling to Guest Class Object

Test Run 2 : Change the execute() to the one shown below:

UserService us=new UserService();
us.setUser(new Admin());

Re-execute the application. The following output is generated.

Figure 7. Tight Coupling to Admin Class Object

In the case of tight-coupling, all the objects along with its all dependencies are created in the code. In contrast to this in the case of loose coupling, the dependency information is stored in a spring configuration file and when the application requests the container for the object containing some dependencies, the container instantiates the object along with its all dependencies dynamically at runtime. The technique is referred to as "Dependency injection" or Inversion of Control, IoC.

Case 2 : Loose Coupling

Test Run 3 : Implement the execute() method of Login class as shown below:

```
ClassPathXmlApplicationContextappContext=newClassPathX
mlApplicationContext("/WEB-INF/applicationContext.xml");

UserService us= (UserService)
        appContext.getBean("userService");
       returnus.message();
```

The dependency configuration in applicationContext.xml file is as follows:

```
<bean id="guest" class="com.siber.Guest"/>
<bean id="admin" class="com.siber.Admin"/>
<bean id="userService" class="com.siber.UserService">
<property name="user">
<ref bean="guest"/>
</property>
</bean>
```

Execute the application. The same output as shown in Figure 6 is generated.

Test Run 4 :Without distrubing the execute() method of Login class, Change the statement shown in bold in applicationContext.xml given above, to the one given below:
<ref bean="admin"/>
and re-execute the application. The same output as shown in Figure 7 is generated.

V. CONCLUSION

Struts, Hibernate and Spring are three emerging technologies for the design and development of J2EE application. Each technology has its own merits and operates in a specific tier of scalable n-tiered enterprise application. The current work provides the blend of these three vital technologies for bringing the merits offered by them under a common roof to a single application.

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,

[4] http://www.codeproject.com/Articles/1086716/JSP-Custom-Tag-Library-for-Table-Joins-and-Master, 19th March, 2016.

[5] https://www.codeproject.com/Articles/1169961/Custom-Annotation-for-Execution-of-Data-Manipulati

[6] Vijayakumar C, Analysis of the Frame work Standard and the Respective Technologies, - International Journal of Computer Science and Information Technology & Security, Vol. 1, No. 1, October 2011.

[7] Tamal Dey, A Comparative Analysis on Modeling and Implementing with MVC Architecture, nternational Conference on Web Services Computing (ICWSC), Proceedings published by International Journal of Computer Applications® (IJCA), pp. 44-49, 2011.

[8] Praveen Gupta, Prof. M.C. Govil, MVC Design Pattern for the multi framework distributed applications using XML, spring and struts Framework, International Journal on Computer Science and Engineering Vol. 02, No. 04, 2010, 1047-1051

[9] Deepak Kasgar, Harish Chandra Maurya, Integration of Struts & Spring & Hibernate for Enterprise Applications, International Journal of Modern Engineering Research, Vol. 5, Iss.4, pp. 52-59, Apr. 2015.

[10] Neha Munsi, Nidhi Sehrawat, Mahak Jain, International Journal of Computer Science and Mobile Computing, Vol. 3, Issue. 10, pp.853–859, October 2014.

[11] Ankur Bawiskar, Prashant Sawant, Vinayak Kankate, Dr. B.B.Meshram, International Journal of Emerging Technology and Advanced Engineering, Volume 2, Issue 6, pp. 203-210, June 2012.