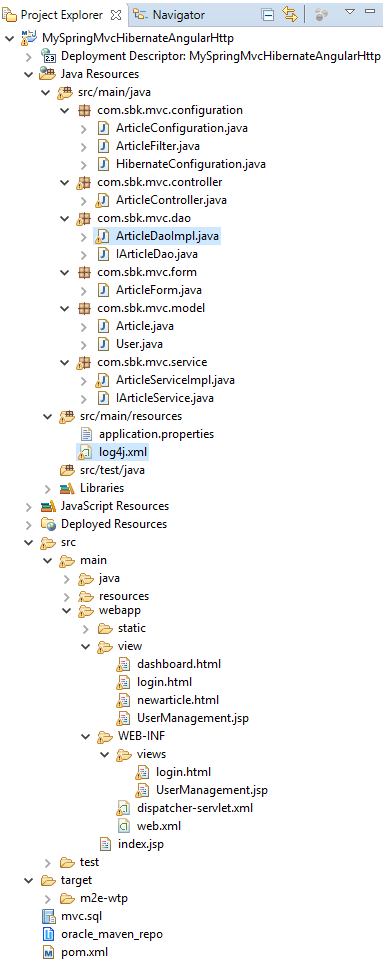
**Faculty Mannual**



This demo demonstrates a complete CRUD application using AngularJS  with $http service to communicate with REST API on server

This is “single page article management app” in which  User can log in into the application to add and view own articles.  
Spring restful services are used by angular js to get the response from DB in JSON format.

In this application below technologies are used

* Java 8, Spring 4 and Hibernate 4
* Angular Js, Html and Bootstrap
* Oracle Db
* Wildfly 9
* Maven as build tool
* Eclipse IDE

Below the code for same

Step 1: create a maven web project and make the changes to pom.xml :

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>com.skuera</groupId>

<artifactId>CMS</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>war</packaging>

<properties>

<maven.compiler.target>1.8</maven.compiler.target>

<maven.compiler.source>1.8</maven.compiler.source>

<springframework.version>4.0.6.RELEASE</springframework.version>

<hibernate.version>4.3.6.Final</hibernate.version>

<mysql.connector.version>5.1.31</mysql.connector.version>

<joda-time.version>2.3</joda-time.version>

<testng.version>6.9.4</testng.version>

<mockito.version>1.10.19</mockito.version>

<h2.version>1.4.187</h2.version>

<dbunit.version>2.2</dbunit.version>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<springframework.version>4.3.1.RELEASE</springframework.version>

<jackson.version>2.7.5</jackson.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${springframework.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-tx</artifactId>

<version>${springframework.version}</version>

</dependency><!-- https://mvnrepository.com/artifact/log4j/log4j -->

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.17</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>${springframework.version}</version>

</dependency>

<!-- Hibernate -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>${hibernate.version}</version>

</dependency>

<!-- jsr303 validation -->

<dependency>

<groupId>javax.validation</groupId>

<artifactId>validation-api</artifactId>

<version>1.1.0.Final</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-validator</artifactId>

<version>5.1.3.Final</version>

</dependency><!-- MySQL -->

<!-- <dependency> <groupId>mysql</groupId> <artifactId>mysql-connector-java</artifactId>

<version>${mysql.connector.version}</version> </dependency> -->

<!-- https://mvnrepository.com/artifact/oracle/ojdbc6 -->

<dependency>

<groupId>com.oracle</groupId>

<artifactId>ojdbc6</artifactId>

<version>11.2.0.3</version>

</dependency>

<!-- Joda-Time -->

<dependency>

<groupId>joda-time</groupId>

<artifactId>joda-time</artifactId>

<version>${joda-time.version}</version>

</dependency>

<!-- To map JodaTime with database type -->

<dependency>

<groupId>org.jadira.usertype</groupId>

<artifactId>usertype.core</artifactId>

<version>3.0.0.CR1</version>

</dependency>

<!-- Servlet+JSP+JSTL -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>3.1.0</version>

</dependency>

<dependency>

<groupId>javax.servlet.jsp</groupId>

<artifactId>javax.servlet.jsp-api</artifactId>

<version>2.3.1</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

<!-- Testing dependencies -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

<version>${springframework.version}</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>${testng.version}</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-all</artifactId>

<version>${mockito.version}</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<version>${h2.version}</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>dbunit</groupId>

<artifactId>dbunit</artifactId>

<version>${dbunit.version}</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>${jackson.version}</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>3.1.0</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

<dependency>

<groupId>commons-fileupload</groupId>

<artifactId>commons-fileupload</artifactId>

<version>1.3.1</version>

</dependency>

<dependency>

<groupId>javax.ws.rs</groupId>

<artifactId>javax.ws.rs-api</artifactId>

<version>2.0-m01</version>

</dependency>

</dependencies>

<build>

<finalName>CMS</finalName>

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.3</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

<artifactId>maven-war-plugin</artifactId>

<version>2.6</version>

<configuration>

<warSourceDirectory>src/main/webapp</warSourceDirectory>

<failOnMissingWebXml>false</failOnMissingWebXml>

</configuration>

</plugin>

</plugins>

</build>

<repositories>

<repository>

<id>codelds</id>

<url>https://code.lds.org/nexus/content/groups/main-repo</url>

</repository>

</repositories>

</project>

2> Step 2: Make changes in 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"* xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"* version=*"3.0"*>

<display-name>MySpringMvcHibernateAngularHttp</display-name>

<welcome-file-list>

<welcome-file>index.html</welcome-file>

</welcome-file-list>

<servlet>

<servlet-name>dispatcher</servlet-name>

<servlet-class>

org.springframework.web.servlet.DispatcherServlet

</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>dispatcher</servlet-name>

<url-pattern>/rest/\*</url-pattern>

</servlet-mapping>

</web-app>

3> Step 3: create dispatcher-servlet.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"*

xmlns:mvc=*"http://www.springframework.org/schema/mvc"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd*

*http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.0.xsd*

*http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.0.xsd*

*http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd"*>

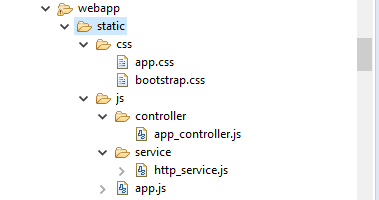
<context:component-scan base-package=*"com.sbk.mvc"*/>

<mvc:annotation-driven/>

</beans>

4> Step 4: copy paste the static folder is webapp

Which contain all the css and js files



5> Now lets configure hibernate properties file create application.properties file in resources folder:

jdbc.driverClassName = oracle.jdbc.driver.OracleDriver

jdbc.url = jdbc:oracle:thin:@localhost:1521:orahome

jdbc.username = system

jdbc.password = system

hibernate.dialect = org.hibernate.dialect.OracleDialect

hibernate.show\_sql = true

hibernate.format\_sql = true



6> Now create log4j.xml in resources folder:

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

<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">

<log4j:configuration debug=*"true"*

xmlns:log4j=*'http://jakarta.apache.org/log4j/'*>

<appender name=*"console"* class=*"org.apache.log4j.ConsoleAppender"*>

<layout class=*"org.apache.log4j.PatternLayout"*>

<param name=*"ConversionPattern"*

value=*"%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1}:%L - %m%n"* />

</layout>

</appender>

<root>

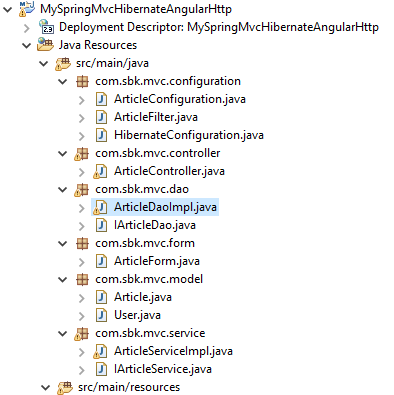
<level value=*"DEBUG"* />

<appender-ref ref=*"console"* />

</root>

</log4j:configuration>

7> Now lets start create our layered architecture as follows



User.java

**package** com.sbk.mvc.model;

**import** java.util.Set;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.OneToMany;

**import** javax.persistence.Table;

/\*\*

\* **@author** Smita

\*/

@Entity

@Table(name = "mvc\_user")

**public** **class** User {

@Id

@GeneratedValue(strategy = GenerationType.***AUTO***)

@Column(name = "user\_id")

**private** **long** id;

@Column(name = "user\_name")

**private** String username;

@Column(name = "user\_pass")

**private** String password;

@OneToMany(mappedBy = "users")

**private** Set<Article> articles;

**public** **long** getId() {

**return** id;

}

**public** **void** setId(**long** 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;

}

}

Article.java

**package** com.sbk.mvc.model;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.JoinColumn;

**import** javax.persistence.ManyToOne;

**import** javax.persistence.Table;

/\*\*

\* **@author** Smita

\*/

@Entity

@Table(name = "mvc\_article")

**public** **class** Article {

@Id

@GeneratedValue(strategy = GenerationType.***AUTO***)

@Column(name = "arc\_id")

**private** **int** articleId;

@Column(name = "arc\_title")

**private** String articleTitle;

@Column(name = "arc\_text")

**private** String articleText;

@Column(name = "img\_path")

**private** String imagePath;

@ManyToOne

@JoinColumn(name = "user\_id", nullable = **false**)

**private** User users;

**public** **int** getArticleId() {

**return** articleId;

}

**public** **void** setArticleId(**int** articleId) {

**this**.articleId = articleId;

}

**public** String getArticleText() {

**return** articleText;

}

**public** **void** setArticleText(String articleText) {

**this**.articleText = articleText;

}

**public** String getImagePath() {

**return** imagePath;

}

**public** **void** setImagePath(String imagePath) {

**this**.imagePath = imagePath;

}

**public** User getUsers() {

**return** users;

}

**public** **void** setUsers(User users) {

**this**.users = users;

}

**public** String getArticleTitle() {

**return** articleTitle;

}

**public** **void** setArticleTitle(String articleTitle) {

**this**.articleTitle = articleTitle;

}

}

ArticleForm.java

**package** com.sbk.mvc.form;

/\*\*

\* **@author** Smita

\*/

**public** **class** ArticleForm {

**private** String articleTitle;

**private** String articleText;

**private** String username;

**public** String getArticleTitle() {

**return** articleTitle;

}

**public** **void** setArticleTitle(String articleTitle) {

**this**.articleTitle = articleTitle;

}

**public** String getArticleText() {

**return** articleText;

}

**public** **void** setArticleText(String articleText) {

**this**.articleText = articleText;

}

**public** String getUsername() {

**return** username;

}

**public** **void** setUsername(String username) {

**this**.username = username;

}

}

IArticleDao.java- Dao Interface

**package** com.sbk.mvc.dao;

**import** java.util.List;

**import** com.sbk.mvc.model.Article;

**import** com.sbk.mvc.model.User;

/\*\*

\* **@author** Smita

\*/

**public** **interface** IArticleDao {

**public** User validate(String user, String pass);

**public** **void** saveArticle(Article article);

**public** User getUserByUserName(String uname);

**public** List<Article> getAllArticle(String username);

**public** Article getArticle(**int** articleId);

}

ArticleDao.java- Dao Impl class

**package** com.sbk.mvc.dao;

**import** java.util.List;

**import** org.hibernate.Query;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Repository;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.sbk.mvc.model.Article;

**import** com.sbk.mvc.model.User;

/\*\*

\* **@author** Smita

\*/

@Repository

@Transactional

**public** **class** ArticleDaoImpl **implements** IArticleDao {

@Autowired

**private** SessionFactory sessionFactory;

@Override

**public** User validate(String user, String pass) {

Session sess = sessionFactory.getCurrentSession();

String hql = "FROM User u where u.username=:uid and u.password=:pass";

Query query = sess.createQuery(hql);

query.setParameter("uid", user);

query.setParameter("pass", pass);

List results = query.list();

User us = **null**;

**if** (results.size() > 0) {

us = (User) results.get(0);

**return** us;

} **else** {

**return** us;

}

}

@Override

**public** **void** saveArticle(Article article) {

Session sess = sessionFactory.getCurrentSession();

System.***out***.println("In Dao");

sess.save(article);

}

@Override

**public** User getUserByUserName(String uname) {

String hql = "from User u where u.username=:uname";

Session sess = sessionFactory.getCurrentSession();

Query qry = sess.createQuery(hql);

System.***out***.println("\*\*\*" + uname);

qry.setString("uname", uname);

User u = (User) qry.uniqueResult();

**return** u;

}

@Override

**public** List<Article> getAllArticle(String username) {

String hql = "from Article a where a.users.username=:uname";

Session sess = sessionFactory.getCurrentSession();

Query qry = sess.createQuery(hql);

System.***out***.println("\*\*\*" + username);

qry.setString("uname", username);

List<Article> alist = qry.list();

System.***out***.println(alist.size());

**return** alist;

}

@Override

**public** Article getArticle(**int** articleId) {

Session sess = sessionFactory.getCurrentSession();

Article art = (Article) sess.load(Article.**class**, articleId);

**return** art;

}

}

**IArticleService.java -Service Interface**

**package** com.sbk.mvc.service;

**import** java.util.List;

**import** com.sbk.mvc.model.Article;

**import** com.sbk.mvc.model.User;

/\*\*

\* **@author** Smita

\*/

**public** **interface** IArticleService {

**public** User isValidUser(String user, String pass);

**public** **int** saveArticle(Article article);

**public** User getUserByUserName(String uname);

**public** List<Article> getAllArticle(String username);

**public** Article getArticle(**int** articleId);

}

**ArticleServiceImpl.java -Service class**

**package** com.sbk.mvc.service;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** com.sbk.mvc.dao.IArticleDao;

**import** com.sbk.mvc.model.Article;

**import** com.sbk.mvc.model.User;

/\*\*

\* **@author** Smita

\*/

@Service("userService")

**public** **class** ArticleServiceImpl **implements** IArticleService {

@Autowired

**private** IArticleDao articleDao;

**public** **void** setDao(IArticleDao articleDao) {

**this**.articleDao = articleDao;

}

@Override

**public** User isValidUser(String user, String pass) {

// **TODO** Auto-generated method stub

**return** articleDao.validate(user, pass);

}

@Override

**public** **int** saveArticle(Article article) {

**int** res = 0;

**try** {

articleDao.saveArticle(article);

res = 1;

} **catch** (Exception e) {

}

**return** res;

}

@Override

**public** User getUserByUserName(String uname) {

// **TODO** Auto-generated method stub

**return** articleDao.getUserByUserName(uname);

}

@Override

**public** List<Article> getAllArticle(String username) {

**return** articleDao.getAllArticle(username);

}

@Override

**public** Article getArticle(**int** articleId) {

**return** articleDao.getArticle(articleId);

}

}

**ArticleController.java -Rest Controller to handle http request**

**package** com.sbk.mvc.controller;

**import** java.util.ArrayList;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.RestController;

**import** com.sbk.mvc.form.ArticleForm;

**import** com.sbk.mvc.model.Article;

**import** com.sbk.mvc.model.User;

**import** com.sbk.mvc.service.IArticleService;

/\*\*

\*

\* **@author** Smita

\*This is spring rest controller to handle http request

\*/

@RestController

**public** **class** ArticleController {

@Autowired

**private** IArticleService articleService;

@RequestMapping(value = "/isvalid", method = RequestMethod.***POST***)

**public** User isValid(@RequestBody User user) {

User us = articleService.isValidUser(user.getUsername(), user.getPassword());

System.***out***.println(us.getUsername());

**return** us;

}

@RequestMapping(value = "/addArticle", method = RequestMethod.***POST***)

**public** Article isValid(@RequestBody ArticleForm article) {

Article art = **new** Article();

art.setArticleText(article.getArticleText());

art.setArticleTitle(article.getArticleTitle());

art.setUsers(articleService.getUserByUserName(article.getUsername()));

**int** res = articleService.saveArticle(art);

**if** (res == 1) {

**return** art;

}

**return** **null**;

}

@RequestMapping(value = "/getAllArticle", method = RequestMethod.***POST***)

**public** List<Article> getArticle(@RequestBody User user) {

List<Article> artList = **new** ArrayList<>();

artList = articleService.getAllArticle(user.getUsername());

**return** artList;

}

@RequestMapping(value = "/getArticle", method = RequestMethod.***POST***)

**public** Article getArticleDetail(@RequestBody Article art) {

Article artcl = **new** Article();

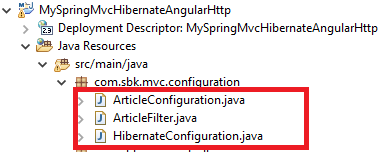
artcl = articleService.getArticle(art.getArticleId());

**return** artcl;

}

}

Lets do the spring boot configuration



ArticleConfiguration.java-Spring boot configuration

**package** com.sbk.mvc.configuration;

**import** org.springframework.context.MessageSource;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.ComponentScan;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.context.support.ResourceBundleMessageSource;

**import** org.springframework.web.servlet.config.annotation.EnableWebMvc;

**import** org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;

**import** org.springframework.web.servlet.config.annotation.ViewResolverRegistry;

**import** org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;

**import** org.springframework.web.servlet.view.InternalResourceViewResolver;

**import** org.springframework.web.servlet.view.JstlView;

/\*\*

\* **@author** Smita

\*

\*/

@Configuration

@EnableWebMvc

@ComponentScan(basePackages = "com.sbk.mvc")

**public** **class** ArticleConfiguration **extends** WebMvcConfigurerAdapter {

@Override

**public** **void** configureViewResolvers(ViewResolverRegistry registry) {

InternalResourceViewResolver viewResolver = **new** InternalResourceViewResolver();

viewResolver.setViewClass(JstlView.**class**);

viewResolver.setPrefix("/WEB-INF/");

viewResolver.setSuffix(".html");

registry.viewResolver(viewResolver);

}

@Bean

**public** MessageSource messageSource() {

ResourceBundleMessageSource messageSource = **new** ResourceBundleMessageSource();

messageSource.setBasename("messages");

**return** messageSource;

}

@Override

**public** **void** addResourceHandlers(ResourceHandlerRegistry registry) {

registry.addResourceHandler("/static/\*\*").addResourceLocations("/static/");

}

}

Creating a Filter – ArticleFilter.java

**package** com.sbk.mvc.configuration;

**import** java.io.IOException;

**import** javax.servlet.Filter;

**import** javax.servlet.FilterChain;

**import** javax.servlet.FilterConfig;

**import** javax.servlet.ServletException;

**import** javax.servlet.ServletRequest;

**import** javax.servlet.ServletResponse;

**import** javax.servlet.http.HttpServletResponse;

/\*\*

\* **@author** Smita

\*

\*/

**public** **class** ArticleFilter **implements** Filter {

**public** **void** doFilter(ServletRequest req, ServletResponse res, FilterChain chain)

**throws** IOException, ServletException {

System.***out***.println("Filtering on..... ");

HttpServletResponse response = (HttpServletResponse) res;

response.setHeader("Access-Control-Allow-Origin", "\*");

response.setHeader("Access-Control-Allow-Methods", "POST, GET, PUT, OPTIONS, DELETE");

response.setHeader("Access-Control-Max-Age", "3600");

response.setHeader("Access-Control-Allow-Headers", "X-requested-with, Content-Type");

chain.doFilter(req, res);

}

**public** **void** init(FilterConfig filterConfig) {

}

**public** **void** destroy() {

}

}

Creating a class for database configuration (Hibernate) – HibernateConfiguration.java