**Knowledge:**

* jdk1.7.0\_51
* Maven 3.1.1
* Tomcat 7.0.26
* Appfuse
* Eclipse Indigo J2EE
* MySQL 5.6
* Toad 7
* GIT
* Notepad++

<http://appfuse.org/display/APF/Using+Spring+MVC>

系统变量：

* Path %JAVA\_HOME%/bin;%JAVA\_HOME%/jre/bin;%MAVEN\_HOME%\bin;%CATALINA\_HOME%/bin;%MYSQL\_HOME%\bin

C:\Apps\ 应用工具软件

C:\projects\ 项目程序

C:\Apps\JAVA\jdk6\jre

* JAVA\_HOME C:\Apps\JAVA\ jdk1.7.0\_51

C:\Apps\ apache-maven-3.0.5

* MAVEN\_HOME C:\Apps\ apache-maven-3.1.1

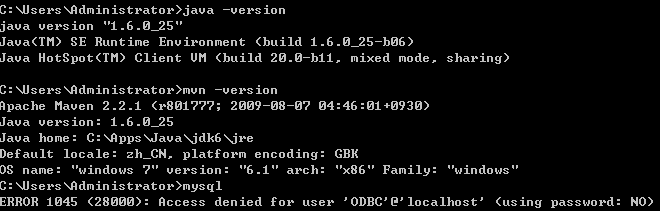
C:\Apps\tomcat\apache-tomcat-7.0.26

C:\Apps\eclipse

* Eclipse Indigo J2EE
* 设定jre目录 C:\Apps\JAVA\jdk6
* 新建java项目 tiebaTool

C:\Apps\MySQL5.6

* Username: “”
* Password: “password”
* MYSQL\_HOME C:\Apps\MySQL5.6\MySQL Server 5.6



pgAdmin III中PostgreSQL8.4——Database——新建数据库LGFAQ

mvn -Dmaven.test.skip=true archetype:generate -B -DarchetypeGroupId=org.appfuse.archetypes -DarchetypeArtifactId=appfuse-basic-spring-archetype -DarchetypeVersion=3.0.0 -DgroupId=com.tiebaTool -DartifactId=myproject -DarchetypeRepository=http://oss.sonatype.org/content/repositories/appfuse

C:\projects>mvn -Ppostgresql -Dmaven.test.skip=true archetype:generate -B -DarchetypeGroupId=org.appfuse.archetypes -DarchetypeArtifactId=appfuse-basic-spring-archetype -DarchetypeVersion=2.1.0 -DgroupId=cn.legendguitar -DartifactId=LGFAQ -DarchetypeRepository=http://oss.sonatype.org/content/repositories/appfuse

C:\projects\LGFAQ>mvn -Ppostgresql -Dmaven.test.skip=true

C:\projects\LGFAQ\pom.xml: 更改为： <dbunit.dataTypeFactoryName>org.dbunit.ext.postgresql.PostgresqlDataTypeFactory</dbunit.dataTypeFactoryName>

<dbunit.operation.type>CLEAN\_INSERT</dbunit.operation.type> <hibernate.dialect>org.hibernate.dialect.PostgreSQLDialect</hibernate.dialect>

<jdbc.groupId>postgresql</jdbc.groupId>

<jdbc.artifactId>postgresql</jdbc.artifactId>

<jdbc.version>8.1-407.jdbc3</jdbc.version>

<jdbc.driverClassName>org.postgresql.Driver</jdbc.driverClassName>

<jdbc.url>jdbc:postgresql://localhost/LGFAQ</jdbc.url>

<jdbc.username>postgres</jdbc.username>

<jdbc.password>postgres</jdbc.password>

If PostgreSQL 9.1:

<version>9.1dev-900.jdbc4</version>

<version>[Version 9.0-801](http://jdbc.postgresql.org/changes.html#version_9.0-801)</version>

C:\projects\LGFAQ> **mvn appfuse:full-source** 全部代码

C:\projects\LGFAQ>mvn jetty:run

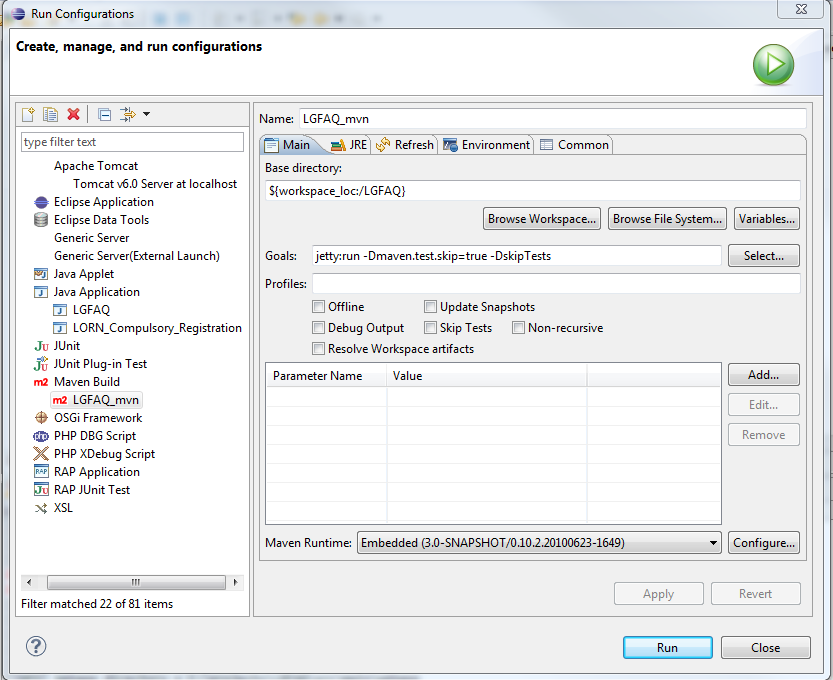
or

C:\projects\LGFAQ>mvn jetty:run-war -Dmaven.test.skip=true 将建立LGFAQ/src/test/resources/sample-data.xml中的所有内容

Or 不运行任何test，保留已创建的table

C:\projects\LGFAQ>mvn jetty:run -Dmaven.test.skip=true -DskipTests

* **Eclipse中运行项目：**



* **登陆页面**  <http://localhost:8080/>

打开 paAdmin

C:\apps\TortoiseSVN

* 右键TortoiseSVN——在C:\projects\ 新建Repository——C:\projects\Repository
* 右键C:\projects\ LGFAQ ——TortoiseSVN——Import
* Eclipse中 Help——install new software——<http://subclipse.tigris.org/update_1.8.x>

XML文件

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

头前不要有任何空行或空格，否则会报错：*org.xml.sax.SAXParseException: The processing instruction target matching "[xX][mM][lL]" is not allowed.*

在web.xml中更改页面整体样式：

<!-- Define the default CSS Theme -->

<context-param>

<param-name>csstheme</param-name>

<param-value>simplicity</param-value>

<!-- param-value>andreas01</param-value> -->

<!-- param-value>puzzlewithstyle</param-value> -->

</context-param>

1)Create a new Model:

*cn.legendguitar.model.Keyword*

2) Register it in “hibernate.cfg.xml”

<hibernate-configuration>

<session-factory>

...

<mapping class=*"cn.legendguitar.model.Keyword"*/>

</session-factory>

</hibernate-configuration>

3) Add new DAO bean in “applicationContext.xml”

<bean id=*"keywordDao"* class=*"cn.legendguitar.dao.hibernate.GenericDaoHibernate"*>

<constructor-arg value=*"cn.legendguitar.model.Keyword"*/>

</bean>

4) Add new manager bean in “applicationContext.xml”

<bean id=*"keywordManager"* class=*"cn.legendguitar.service.impl.GenericManagerImpl"*>

<constructor-arg ref=*"keywordDao"*/>

</bean>

5) Create new controller:

cn.legendguitar.webapp.controller.KeywordsController

6) Create new .jsp:

/LGFAQ/src/main/webapp/WEB-INF/pages/admin/keywords.jsp

7) Add ~ to LGFAQ-servlet.xml

<bean id=*"keywordController"* class=*"cn.legendguitar.webapp.controller.KeywordController"*>

<property name=*"keywordManager"* ref=*"keywordManager"*/>

<property name=*"roleManager"* ref=*"roleManager"*/>

</bean>

* **Display Table tag:**

Keyword.java

**public** **class** KeywordController {

@RequestMapping(method = RequestMethod.*GET*)

**public** ModelAndView handleRequest() **throws** Exception {

Map<String, Object> map = **new** HashMap<String, Object>();

map.put("allKeywords", keywordManager.getAll());

map.put("allRoles", roleManager.getAll());

**return** **new** ModelAndView("/admin/keywords",map);

}

keywords.jsp

<display:table name="allKeywords" **class**="table" requestURI="/admin/keywords" id="keywordList" **export**="true" pagesize="40" defaultsort="1" defaultorder="ascending">

<display:column property="id" sortable="true" href="keywordform" media="html" paramId="id" paramProperty="id" titleKey="keyword.id"/>

<display:column property="id" media="csv excel xml pdf" titleKey="keyword.id"/>

<display:column property="word" sortable="true" titleKey="keyword.word"/>

<display:column property="usedTimes" sortable="true" titleKey="keyword.usedTimes"/>

<display:column property="createdBy" sortable="true" titleKey="keyword.createdBy"/>

<display:column property="timeCreated" sortable="true" titleKey="keyword.timeCreated"/>

<display:column property="modifiedBy" sortable="true" titleKey="keyword.modifiedBy"/>

<display:column property="timeLastModified" sortable="true" titleKey="keyword.timeLastModified"/>

<display:setProperty name="paging.banner.item\_name"><fmt:message key="keywordList.keyword"/></display:setProperty>

<display:setProperty name="paging.banner.items\_name"><fmt:message key="keywordList.keywords"/></display:setProperty>

<display:setProperty name="export.excel.filename"><fmt:message key="keywordList.title"/>.xls</display:setProperty>

<display:setProperty name="export.csv.filename"><fmt:message key="keywordList.title"/>.csv</display:setProperty>

<display:setProperty name="export.pdf.filename"><fmt:message key="keywordList.title"/>.pdf</display:setProperty>

<display:setProperty name="basic.show.header" value="true" />

<display:setProperty name="sort.amount" value="list" />

</display:table>

* **不需要Controller的jsp页面，设置在×××-servelt.xml的这里：**

<mvc:view-controller path=*"/mainMenu"* view-name=*"mainMenu"*/>

<mvc:view-controller path=*"/help"* view-name=*"help"*/>

* **Checkboxes只适用于传递对象为String的情况：**

Faq faq = faqManager.get(**new** BigInteger(id));

List<Keyword> keywordList = **new** ArrayList<Keyword>();

List<String> keywordWordList = **new** ArrayList<String>();

Set<Keyword> keywordSet = faq.getKeywords();

**for** (Keyword keyword : keywordSet) {

keywordList.add(keyword);

keywordWordList.add(keyword.getWord());

}

request.setAttribute("keywordList", keywordList);

request.setAttribute("keywordWordList", keywordWordList);

**return** faq;

<form:checkboxes items="${keywordWordList}" path="keywords" />

electricacoustic弗拉门戈finglestyleclassic

* **在String onSubmin中跳转用”redirect:url”：**

**public** String onSubmit(Faq faq, BindingResult errors, HttpServletRequest request,

HttpServletResponse response)

**throws** Exception {

**……**

**return** "redirect:/admin/keywords";

* **showForm的基本形式：**

@ModelAttribute

@RequestMapping(method = {RequestMethod.*GET*, RequestMethod.*POST*})

**protected** ModelAndView showForm(HttpServletRequest request)

**throws** Exception {

Map<String, Object> map = **new** HashMap<String, Object>();

map.put("isAdmin", util.isCurrentUserAdmin());

// map.put("allKeywords", keywordManager.getAll()); 与下句功能同；

request.setAttribute("allKeywords", keywordManager.getAll());

**return** **new** ModelAndView("/admin/keywords",map);

}

* Pom.xml文件Dependency构成

<http://www.360doc.com/content/11/0616/19/1947337_127426214.shtml>



<dependency>

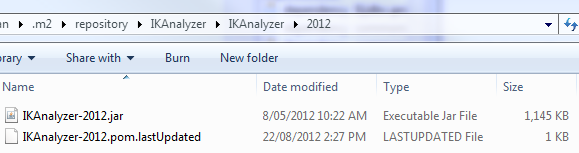
<groupId>IKAnalyzer</groupId>

<artifactId>IKAnalyzer</artifactId>

<version>2012</version>

</dependency>

Run mvn eclipse:eclipse



* **双向MANY TO MANY的声明写法:**

@ManyToMany(cascade={CascadeType.*PERSIST*, CascadeType.*REFRESH*},fetch = FetchType.*LAZY*)

@JoinTable(

name = "faq\_keyword",

joinColumns = { @JoinColumn(name = "keyword\_id") },

inverseJoinColumns = @JoinColumn(name = "faq\_id")

)

**public** List<Faq> getFaqs() {

**return** faqs;

}

...

@ManyToMany(cascade={CascadeType.*PERSIST*, CascadeType.*REFRESH*},fetch = FetchType.*EAGER*)

@JoinTable(

name = "faq\_keyword",

joinColumns = { @JoinColumn(name = "faq\_id") },

inverseJoinColumns = @JoinColumn(name = "keyword\_id")

)

**public** List<Keyword> getKeywords() {

**return** keywords;

}

...

1. cascade的值只能从CascadeType.PERSIST（级联新建）、CascadeType.REMOVE（级联删除）、 CascadeType.REFRESH（级联刷新）、CascadeType.MERGE（级联更新）中选择一个或多个。还有一个选择是使用 CascadeType.ALL，表示选择全部四项。
2. 当CascadeType为ALL或REMOVE时，如果删除一个Keyword，将不光删除connection of Keyword and Faq from the connection table, 还会删除曾引用该Keyword的所有Faq。慎用！
3. 两DAO互为joinColumns 和inverseJoinColumns
4. 当Faq方的fetch = FetchType.*LAZY* 时，出现异常：

*org.hibernate.LazyInitializationException: failed to lazily initialize a collection of role*

并无法通过getter取到Keyword集合

1. FetchType.*LAZY比*FetchType.*EAGER*效率高，不全加载。
2. 选用Set适用于无序无重复的集合，选择List适用于有序有重复的集合；

* **判断当前页面名**

<div id="page" **class**='<%= request.getRequestURI().substring(1) %>'>

<div id="page" **class**=<%= request.getRequestURI().substring(request.getRequestURI().lastIndexOf("/")+1) %>'>

* **返回当前用户**

${pageContext.request.remoteUser}

${pageContext.request.userPrincipal.name}

* **返回当前用户使用语言**

${pageContext.request.locale.language}

* **判断当前用户角色，并在JSTL中调用**

Jsp中的java变量只有在被存进request.session.attribute或request.attribute中后，才能被JSTL调用！

<%

String userRole = **null**;

**if** (request.isUserInRole("ROLE\_OWNER")) {

userRole = "ROLE\_OWNER";

}

request.getSession().setAttribute("userRole", userRole);

%>

<c:choose>

<c:when test='${not empty userRole}'>

<fmt:message key="your\_role\_is"/><fmt:message key='<%=userRole%>'/>(<%=userRole%>)

</c:when>

<c:otherwise>

<fmt:message key="please\_login"/>

</c:otherwise>

</c:choose>

* **解决css文件被Jetty锁定无法在Eclipse中更改的问题：**

C:\Users\sbao\.m2\repository\org\mortbay\jetty\jetty\6.1.26\ jetty-6.1.26.jar

C:\Users\sbao\.m2\repository\org\mortbay\jetty\jetty\6.1.26\jetty-6.1.26\org\mortbay\jetty\webapp\webdefault.xml

<http://rongjih.blog.163.com/blog/static/335744612010620341622/>

<http://dengyin2000.iteye.com/blog/246774>

* **Message可以使用jsp中java变量**

<fmt:message key='<%=userRole%>'/>

* **displayTag和错误提示中文乱码的解决办法**

在pom.xml中添加

<native2ascii src="web/WEB-INF/classes" dest="${build.dir}/web/classes"   
                includes="displaytag\_zh\*.properties" encoding="UTF-8"/>

* **在UserDaoHibernate.java中无法将Biginteger直接写进sql query，应该先转化成long**

**public** String getUserPasswordById(BigInteger id) {

SimpleJdbcTemplate jdbcTemplate =

**new** SimpleJdbcTemplate(SessionFactoryUtils.*getDataSource*(getSessionFactory()));

Table table = AnnotationUtils.*findAnnotation*(User.**class**, Table.**class**);

**return** jdbcTemplate.queryForObject("select password from " + table.name() + " where id=?", String.**class**, id.longValue());

}

* **将BigInteger转化成String？**

**new** String(user.getId().toByteArray())

* **屏蔽信息显示：**

WARN [24470614@qtp-13797435-0] ValidatorResources.getForm(448) | Form 'faq' not found for locale 'zh\_CN'

WARN [24470614@qtp-13797435-0] ValidatorResources.getForm(448) | Form 'null' not found for locale 'zh\_CN'

解决：在validation.xml中添加：

<formset>

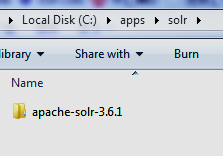
<form name=*"faq"* />

</formset>

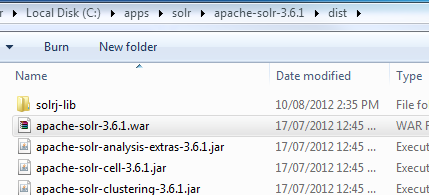
* 去掉字符串中的回车换行

s = s.replaceAll("\r\n|\r|\n", "");

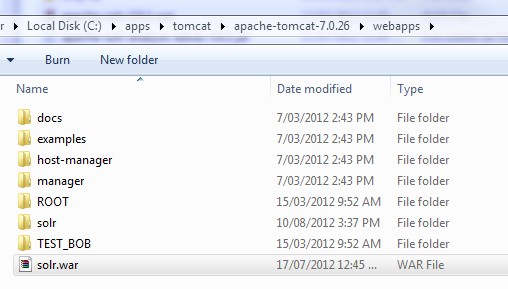
* **配置solr**



1. Download solr and decompress .zip to dir: C:\\apps\solr\



1. Copy .war to tomcat-xx\webapps\, then change name to solr.war



1. Create file “solr.xml” under C:\apps\tomcat\apache-tomcat-7.0.26\conf\Catalina\localhost\

<Context docBase="C:/apps/tomcat/apache-tomcat-7.0.26/webapps/solr.war" reloadable="true" >

<Environment name="solr/home" type="java.lang.String" value="C:/apps/solr/apache-solr-3.6.1/example/solr" override="true" />

</Context>

1. Run tomcat in cmd:

C:\apps\tomcat\apache-tomcat-7.0.26\bin\startup.bat

1. Go link: <http://localhost:8081/solr/admin/> (if tomcat is running at port 8081)

* **中文分词IKAnalyzer**

1. 下载IKAnalyzer2012\_u5.zip
2. Copy ‘IKAnalyzer2012.jar’ to “C:\apps\tomcat\apache-tomcat-7.0.26\webapps\solr\WEB-INF\lib\”
3. Backup and update “C:\apps\solr\apache-solr-3.6.1\example\solr\conf\schema.xml”:

Add:

<!-- IKAnalyzer3.2.8 中文分词-->

<fieldType name="text" class="solr.TextField">

<analyzer class="org.wltea.analyzer.lucene.IKAnalyzer" />

<analyzer type="index">

<tokenizer class="org.wltea.analyzer.solr.IKTokenizerFactory" isMaxWordLength="false" />

<filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" />

<filter class="solr.WordDelimiterFilterFactory" generateWordParts="1" generateNumberParts="1" catenateWords="1" catenateNumbers="1" catenateAll="0" splitOnCaseChange="1" />

<filter class="solr.LowerCaseFilterFactory" />

<filter class="solr.EnglishPorterFilterFactory" protected="protwords.txt" />

<filter class="solr.RemoveDuplicatesTokenFilterFactory" />

</analyzer>

<analyzer type="query">

<tokenizer class="org.wltea.analyzer.solr.IKTokenizerFactory" isMaxWordLength="true" />

<filter class="solr.StopFilterFactory" ignoreCase="true" words="stopwords.txt" />

<filter class="solr.WordDelimiterFilterFactory" generateWordParts="1" generateNumberParts="1" catenateWords="1" catenateNumbers="1" catenateAll="0" splitOnCaseChange="1" />

<filter class="solr.LowerCaseFilterFactory" />

<filter class="solr.EnglishPorterFilterFactory" protected="protwords.txt" />

<filter class="solr.RemoveDuplicatesTokenFilterFactory" />

</analyzer>

</fieldType>

To <types> …… </types>

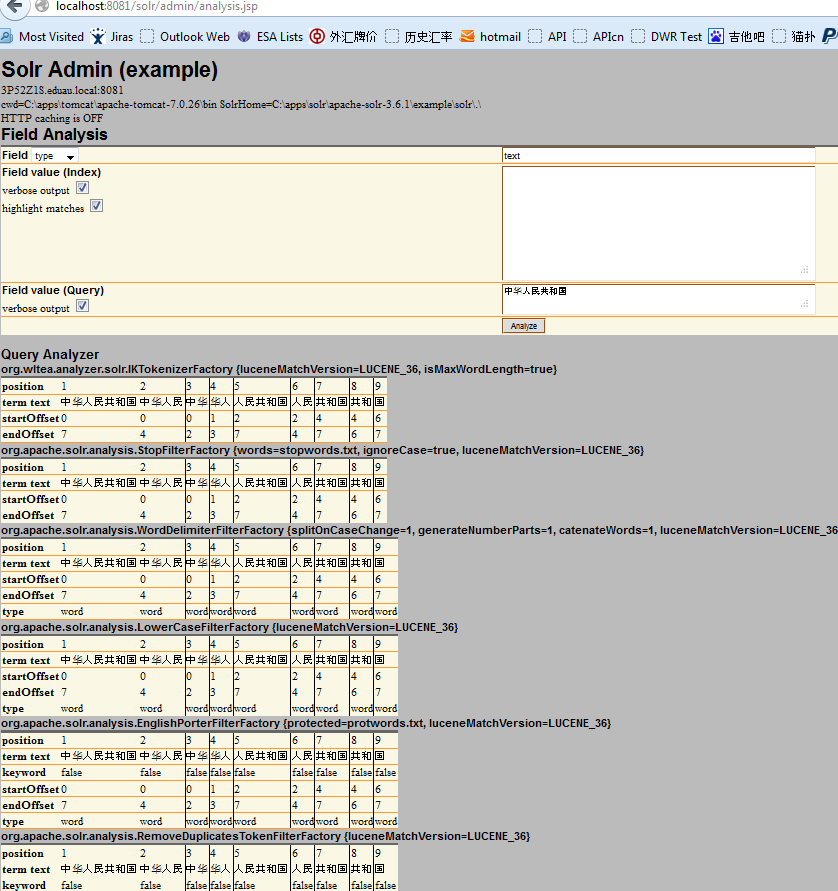
Add:

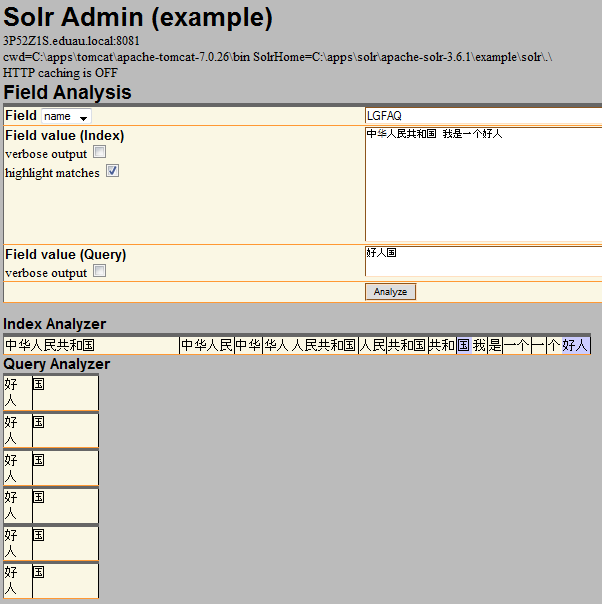
<field name="name" type="text" indexed="true" stored="true" required="true" />

And comment off:

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

1. Restart tomcat and check <http://localhost:8081/solr/admin/analysis.jsp>





* Java heap space problem



To solve, just reset parameter in “C:\apps\tomcat\apache-tomcat-7.0.26\bin\catalina.bat”

set JAVA\_OPTS=-Xms512m -Xmx1024m -XX:PermSize=256M -XX:MaxNewSize=512m -XX:MaxPermSize=512m

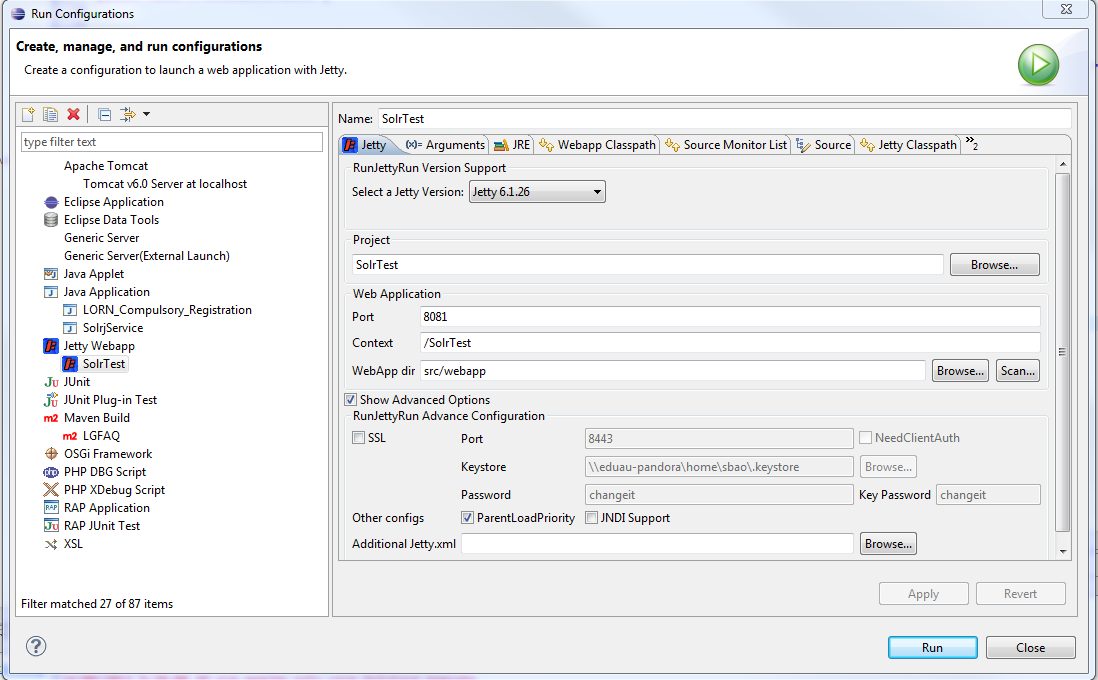
* Setup Solr app within Eclipse

1. Install jetty plugin to Eclipse:

<http://code.google.com/p/run-jetty-run/wiki/GettingStarted>

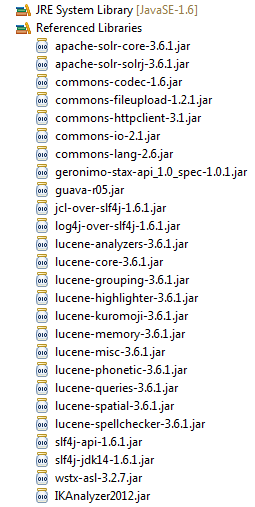
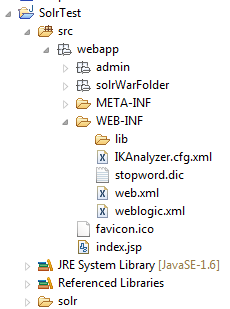
Help -> Updates -> Find and install... -> Search for new features to install

<http://run-jetty-run.googlecode.com/svn/trunk/updatesite>



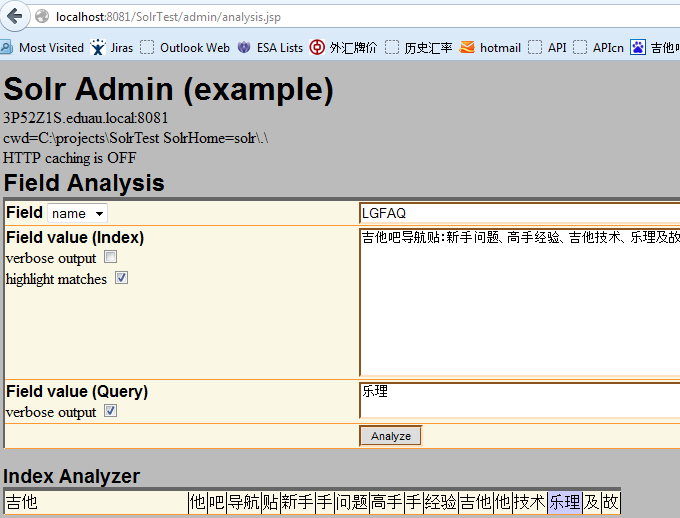
<http://hokiesuns.blogspot.com.au/2010/01/setting-up-apache-solr-in-eclipse.html>

1. Create new Solr app and setup IKAnalyzer:



1. Run Solr app in Eclipse and access admin page:

<http://localhost:8081/SolrTest/admin/analysis.jsp>



* 往项目中加入新jar

把.jar放入” C:\projects\LGFAQ\src\main\webapp\WEB-INF\lib\”，右键Build Path -> add to build path

* 连接Posrtgresql数据库并读入数据

1. Solrconfig.xml

<requestHandler name=*"/dataimport"* lass=*"org.apache.solr.handler.dataimport.DataImportHandler"*>

<lst name=*"defaults"*>

<str name=*"config"*>data-config.xml</str>

</lst>

</requestHandler>

2) data-config.xml

<dataConfig>

<dataSource name=*"JdbcDataSource"*

driver=*"org.postgresql.Driver"*

url=*"jdbc:postgresql://localhost/LGFAQ"*

user=*"postgres"*

password=*"postgres"*/>

<document>

<entity name=*"id"* query=*"select id, title, content, author, link, remark from faq"* >

<field column=*"id"* name=*"id"*/>

<field column=*"title"* name=*"title"*/>

<field column=*"content"* name=*"content"*/>

<field column=*"author"* name=*"author"*/>

<field column=*"link"* name=*"link"*/>

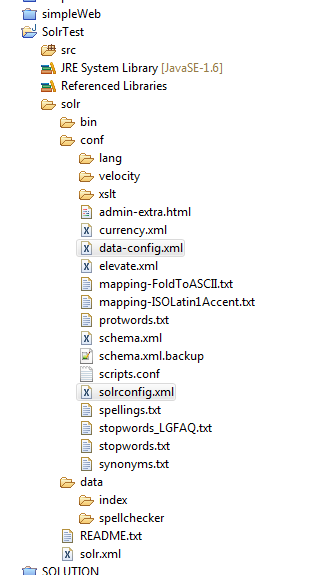
<field column=*"remark"* name=*"remark"*/>

</entity>

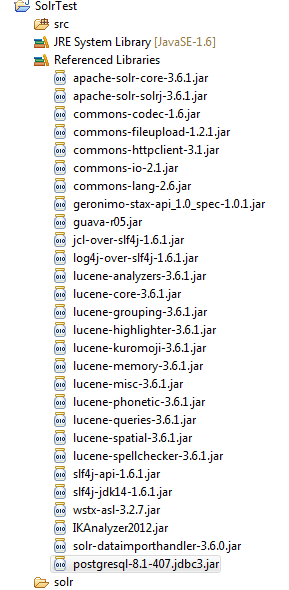
</document>

</dataConfig>

3) 放置xml位置



4） add进postgresql jdbc drive jar



5） 运行<http://localhost:8081/SolrTest/dataimport?command=full-import> 从数据库中全载入数据

6） 查看数据[http://localhost:8081/SolrTest/select/?q=\*%3A\*&version=2.2&start=0&rows=10&indent=on](http://localhost:8081/SolrTest/select/?q=*%3A*&version=2.2&start=0&rows=10&indent=on)

* 在solr中插入新field

SolrTest/solr/conf/data/config.xml

<document>

<entity name=*"id"* query=*"select id, title, content, author, link, remark from faq"* >

<field column=*"id"* name=*"id"*/>

<field column=*"title"* name=*"title"*/>

<field column=*"content"* name=*"content"*/>

<field column=*"text"* name=*"text"*/>

<field column=*"author"* name=*"author"*/>

<field column=*"link"* name=*"link"*/>

<field column=*"remark"* name=*"remark"*/>

</entity>

</document>

cn.legendguitar.serviceSolrIndexingService

**public** **synchronized** **void** startPolling() **throws** Exception {

……

List<String> fieldList = **new** ArrayList<String>();

fieldList.add("title");

fieldList.add("author");

fieldList.add("content");

fieldList.add("text");

fieldList.add("link");

fieldList.add("remark");

cn.legendguitar.model.SolrFaqObject

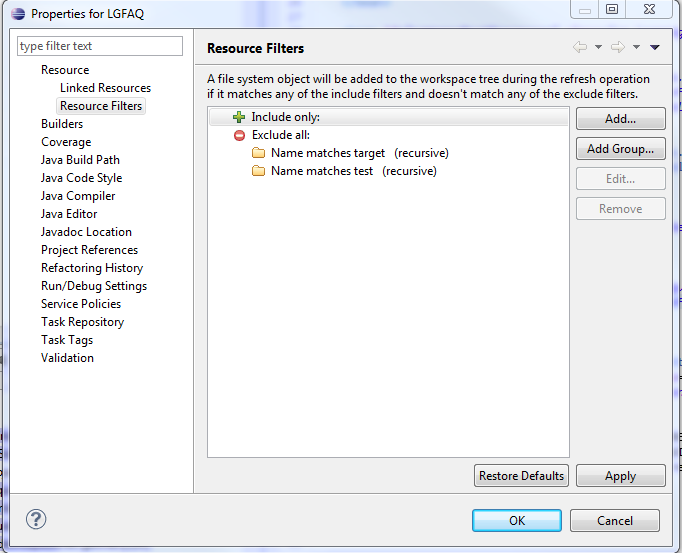
@Field(value = "text")

**private** String text;

getter & setter

* 如何在Eclipse中指定查找范围scope（可以加快search速度）

1. Go to Project -> Properties -> Resource Filters.
2. Click *Add*.
3. Choose Filter type: Exclude all
4. Choose Applies to: Folders; check All children (recursive).
5. In the Attributes, choose *Name*, *Matches*, .svn (for example).



* 如何在Spring中定时运行Solr as Scheduled task

1. 在/LGFAQ/src/main/webapp/WEB-INF/web.xml中

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>

classpath:/applicationContext-resources.xml

classpath:/applicationContext-dao.xml

classpath:/applicationContext-service.xml

classpath:/applicationContext-job.xml

classpath\*:/applicationContext.xml

/WEB-INF/applicationContext\*.xml

/WEB-INF/cxf-servlet.xml

/WEB-INF/security.xml

/WEB-INF/LGFAQ-servlet.xml

</param-value>

</context-param>

1. 在/LGFAQ/src/main/resources/applicationContext-job.xml中

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

<beans xmlns=*"http://www.springframework.org/schema/beans"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"*

*http://www.springframework.org/schema/beans*

[*http://www.springframework.org/schema/beans/spring-beans-2.5.xsd*](http://www.springframework.org/schema/beans/spring-beans-2.5.xsd)*"*>

<bean id=*"solrIndexingService"* class=*"cn.legendguitar.service.SolrIndexingService"*>

<property name=*"faqDao"* ref=*"faqDao"* />

<property name=*"solrIndexRequestDao"* ref=*"solrIndexRequestDao"* />

<property name=*"solrUrl"* value=*"${solr.serverUrl}"* />

</bean>

<bean id=*"schedulerTask"* class=*"org.springframework.scheduling.timer.MethodInvokingTimerTaskFactoryBean"*>

<property name=*"targetObject"* ref=*"solrIndexingService"* />

<property name=*"targetMethod"* value=*"startPolling"* />

</bean>

<bean id=*"timerTask"* class=*"org.springframework.scheduling.timer.ScheduledTimerTask"*>

<!-- property name="delay" value="${solrIndexingServiceStartPollingDelay}" /> Not working. Unable to convert String to Long type-->

<property name=*"timerTask"* ref=*"schedulerTask"* />

<property name=*"delay"* value=*"2000"* /> <!—延时2秒钟 -->

<property name=*"period"* value=*"300000"* /> <!—每5分钟循环运行一次 -->

</bean>

<!-- start solrIndexingService polling for solr index requests in its own thread -->

<bean class=*"org.springframework.scheduling.timer.TimerFactoryBean"*>

<property name=*"scheduledTimerTasks"*>

<list>

<ref local=*"timerTask"* />

</list>

</property>

</bean>

</beans>

1. 在/LGFAQ/src/main/resources/ApplicationResources.properties中

# About solr service;

solr.serverUrl=http://localhost:8081/solr/

1. 每隔5分钟系统将自动运行

/LGFAQ/src/main/java/cn/legendguitar/service/SolrIndexingService.java的startPolling()方法

* 加入新菜单Search

修改menu.jsp

修改menu-config.xml

修改search.jsp中的

<head>

<title><fmt:message key="search.title"/></title>

<meta name="heading" content="<fmt:message key='search.heading'/>"/>

<meta name="menu" content="Search"/>

</head>

使得在访问该页时相应菜单高亮

* 抓取网页内容用com.gargoylesoftware.htmlunit

WebClient webClient = **new** WebClient(BrowserVersion.*FIREFOX\_3\_6*);

URL url = **new** URL("http://tieba.baidu.com/f/good?kw=%BC%AA%CB%FB");

HtmlPage page = (HtmlPage) webClient.getPage(url);

* 抓取<ul>下的所有<li>

List<?> lzl\_cntList = d\_post\_replydiv.getByXPath(".//ul[@class='j\_lzl\_m\_w']/li");

**for** (Object liObj : lzl\_cntList) {

HtmlListItem j\_lzl\_s\_p = (HtmlListItem) liObj;

HtmlAnchor lzl\_cntAuthorA = (HtmlAnchor) j\_lzl\_s\_p.getFirstByXPath(".//a[@class='at']");

* 如果出现 Cookie rejected 错误：

Cookie rejected: "[version: 0][name: xxxxx][value: 5][domain: .xxx.xx][path: /][expiry: Tue Feb 21 08:53:46 MSK 2013]". Illegal domain attribute "xxxx.xx”

可以开个FireFox去该页面，然后取出相应cookie值

WebClient webClient = **new** WebClient(BrowserVersion.*INTERNET\_EXPLORER\_8*);

webClient.getCookieManager().setCookiesEnabled(**true**);

Cookie cookie = **new** Cookie(".baidu.com", "BAIDUID", "6285C6853F5A7F9E2CC2F41DA9DCFE53:FG=1");

webClient.getCookieManager().addCookie(cookie);

* Utf-8内容如果不带bom头字节，输出在网页上会出现乱码

解决：手动添加三段字节码

**private** **void** outputString(String content, String outputFilePath) {

**try** {

FileOutputStream fos = **new** FileOutputStream(outputFilePath);

**byte** [] data = **new** **byte**[] { (**byte**)0xEF, (**byte**)0xBB, (**byte**)0xBF };

fos.write(data);

Writer out = **new** OutputStreamWriter(fos, "UTF-8");

out.write(content);

out.close();

} **catch** (FileNotFoundException e) {

e.printStackTrace();

} **catch** (IOException e) {

e.printStackTrace();

}

}

* 在search.jsp页做输入关键词分离和内容中关键词高亮

用javascript，见search.jsp

* 将HTML标签实体转回正常标签，decode/unescape

**function** encodeAllHtmlCode(instr){

**return** $('<div/>').html(instr).text();

}

* 无视大小写的字符串全替换

result = result.replaceAll("(?i)"+word, "<span class='highlighted'>" + word + "</span>");

* 对HashMap的操作

1. 定义：

Map<Integer, String> indexWordMap = **new** HashMap<Integer, String>();

1. 赋值：

indexWordMap.put(index, word);

1. 循环中取值：

**for** (Map.Entry<Integer, String> entry : indexWordMap.entrySet()) {

**int** wordIndex = (Integer) entry.getKey();

**int** wordLength = ((String) entry.getValue()).length();

}

1. 按Key（或Value）排序：

Map<Integer, String> sortedIndexWordMap = sortByComparator(indexWordMap);

//sort a Map (HashMap) based on its keys from greatest index to smallest index;

**private** Map<Integer, String> sortByComparator(Map<Integer, String> unsortMap) {

List<Integer> list = **new** LinkedList(unsortMap.entrySet());

Collections.*sort*(list, **new** Comparator() {

**public** **int** compare(Object o1, Object o2) {

**return** ((Comparable) ((Map.Entry) (o1)).getKey()).compareTo(((Map.Entry) (o2)).getKey()) \* -1;

}

});

// put sorted list into map again

Map sortedMap = **new** LinkedHashMap();

**for** (Iterator it = list.iterator(); it.hasNext();) {

Map.Entry entry = (Map.Entry) it.next();

sortedMap.put(entry.getKey(), entry.getValue());

}

**return** sortedMap;

}

* 破解百度的图片链接防盗链机制

思路：下载所有图片至本地服务器，更换页面图片链接

**private** String localizePics(String divContent, String localDir) {

// http://imgsrc.baidu.com/forum/pic/item/6a63f6246b600c33d84501511a4c510fd9f9a16f.jpg

String tempContent = divContent;

String imageUrlPath = "http://imgsrc.baidu.com/forum/pic/item/";

String imageServerPath = "/downloadPics/pic/item/";

**int** imageStartIndex = tempContent.indexOf(imageUrlPath);

**if** (imageStartIndex >= 0) {

**int** imageEndIndex = tempContent.indexOf("\"", imageStartIndex);

String imageUrlStr = tempContent.substring(imageStartIndex, imageEndIndex);

String imageName = imageUrlStr.substring(imageUrlPath.length());

**try** {

downloadAndStoreImage(imageUrlStr, localDir, imageName);

divContent = divContent.replaceAll("http://imgsrc.baidu.com/forum/pic/item/", imageServerPath);

} **catch** (Exception e) {

e.printStackTrace();

}

}

**return** divContent;

}

**private** **void** downloadAndStoreImage(String imageUrlStr, String localDir, String newImageFileName) **throws** Exception {

// 构造URL

URL url = **new** URL(imageUrlStr);

// 打开URL连接

URLConnection con = url.openConnection();

// 得到URL的输入流

InputStream input = con.getInputStream();

// 设置数据缓冲

**byte**[] bs = **new** **byte**[1024 \* 2];

// 读取到的数据长度

**int** len;

// 输出的文件流保存图片至本地

OutputStream os = **null**;

**try** {

String picFullPath = localDir + newImageFileName;

**if** (! **new** File(picFullPath).exists()) {

os = **new** FileOutputStream(localDir + newImageFileName);

**while** ((len = input.read(bs)) != -1) {

os.write(bs, 0, len);

}

}

} **catch** (FileNotFoundException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

os.close();

input.close();

}

* 在服务器目录里添加新图片文件夹，并将图片显示于页面

1）security.xml

<http auto-config=*"true"* lowercase-comparisons=*"false"*>

<intercept-url pattern=*"/downloadPics/\*\*"* filters=*"none"*/>

<intercept-url pattern=*"/images/\*\*"* filters=*"none"*/>

2) urlrewrit.xml

<rule>

<from>/downloadPics/\*\*</from>

<to>/downloadPics/$1</to>

</rule>

<rule>

<from>/images/\*\*</from>

<to>/images/$1</to>

</rule>

1. 图片链接为：/downloadPics/pic/item/d043ad4bd11373f00306372ba40f4bfbfbed044c.jpg

* 更改网站整体css样式和大标题

Web.xml

<display-name>appfuse</display-name>

<distributable/>

<!-- precompiled jsp mappings -->

<!-- Define the default CSS Theme -->

<context-param>

<param-name>csstheme</param-name>

<param-value>simplicity</param-value>

<!-- param-value>andreas01</param-value-->

<!-- param-value>puzzlewithstyle</param-value-->

ApplicationResources.properties

webapp.name=LegendGuitar FAQ System

#webapp.name=Appfuse

webapp.tagline=Providing guitar beginner with sufficient knowledge and considerate assistance.

# webapp.tagline=Free Tutorial.

* IK Analyzer扩展辞典

/SolrTest/src/IKAnalyzer.cfg.xml

<properties>

<comment>IK Analyzer 扩展配置</comment>

<!--用户可以在这里配置自己的扩展字典 -->

<entry key=*"ext\_dict"*>/webapp/WEB-INF/dic/ext.dic</entry>

<!-- -->

<!--用户可以在这里配置自己的扩展停止词字典-->

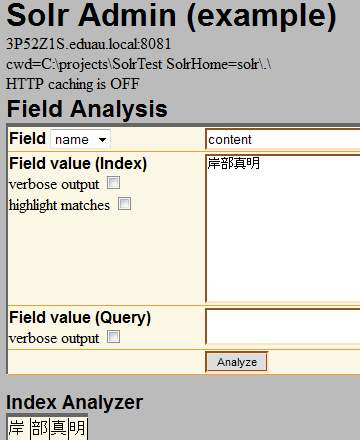
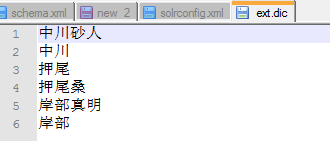
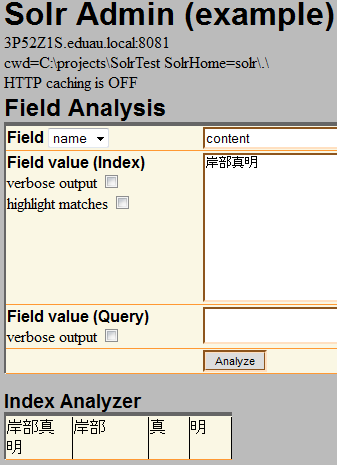
<entry key=*"ext\_stopwords"*>/webapp/WEB-INF/dic/stopword.dic</entry>

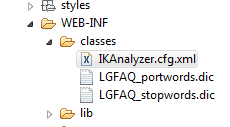
</properties>

有三点要注意（要不然扩展词典始终不生效）：

1. 后缀名.dic的词典文件，必须如使用文档里所说的 无BOM的UTF-8编码保存的文件。如果不确定什么是  无BOM的UTF-8编码，最简单的方式就是 用Notepad++编辑器打开，Encoding->选择 Encoding in UTF-8 without BOM，然后保存。
2. 项目preferences 里 编码选择 utf8。
3. 词典和IKAnalyzer.cfg.xml配置文件的路径问题。**IKAnalyzer.cfg.xml必须在src根目录下。**词典可以任意放，但是在IKAnalyzer.cfg.xml里要配置对。如下：我的两个词典文件my.dic 和 mine.dic 放在src下的com.org.config包下，**注意com前面一定不要加/，否则是绝对路径找不到。**

Before After



在Java项目LGFAQ中，如此放置这三个文件。

（如果没有classes目录要新建）

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

<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">

<properties>

<comment>IK Analyzer 扩展配置</comment>

<!--用户可以在这里配置自己的扩展字典 -->

<entry key=*"ext\_dict"*>/LGFAQ\_portwords.dic;</entry>

<!--用户可以在这里配置自己的扩展停止词字典-->

<entry key=*"ext\_stopwords"*>/LGFAQ\_stopwords.dic;</entry>

</properties>

自定义字典位于：C:\projects\LGFAQ\src\main\webapp\WEB-INF\classes\

* 将项目原用的PostgreSQL DB换成MySQL DB

1. 安装好MySQL （5.1.20）在“C:\apps\MySQL\”目录下；
2. 在Workbench中新建DB “LGFAQ”
3. 使用<http://www.lightbox.ca/pg2mysql.php> pg2mysql工具，把postgreSQL转换成MySQL；
4. 可以先转换和新建所有tables及columns，之后再添加DB内容；
5. LGFAQ/pom.xml 注释掉所有postgreSQL相关行，添加：

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.20</version>

</dependency>

<!-- ================= Database Profiles ================= -->

<profile>

<id>mysql</id>

<activation>

<activeByDefault>true</activeByDefault>

</activation>

<properties>

<hibernate.dialect>org.hibernate.dialect.MySQL5InnoDBDialect</hibernate.dialect>

<jdbc.groupId>mysql</jdbc.groupId>

<jdbc.artifactId>mysql-connector-java</jdbc.artifactId>

<jdbc.version>5.1.20</jdbc.version>

<jdbc.driverClassName>com.mysql.jdbc.Driver</jdbc.driverClassName>

<jdbc.url>jdbc:mysql://localhost/${db.name}?createDatabaseIfNotExist=true&amp;amp;useUnicode=true&amp;amp;characterEncoding=utf-8&amp;amp;autoReconnect=true</jdbc.url>

<jdbc.username>root</jdbc.username>

<jdbc.password/>

</properties>

</profile>

<properties>

……

<!-- Database settings -->

<dbunit.dataTypeFactoryName>org.dbunit.ext.mysql.MySqlDataTypeFactory</dbunit.dataTypeFactoryName>

<dbunit.operation.type>CLEAN\_INSERT</dbunit.operation.type>

<hibernate.dialect>org.hibernate.dialect.MySQL5InnoDBDialect</hibernate.dialect>

<jdbc.groupId>mysql</jdbc.groupId>

<jdbc.artifactId>mysql-connector-java</jdbc.artifactId>

<jdbc.version>5.1.20</jdbc.version>

<jdbc.driverClassName>com.mysql.jdbc.Driver</jdbc.driverClassName>

<jdbc.url>jdbc:mysql://localhost/${db.name}?createDatabaseIfNotExist=true&amp;amp;useUnicode=true&amp;amp;characterEncoding=utf-8&amp;amp;autoReconnect=true</jdbc.url>

<jdbc.username>root</jdbc.username>

<jdbc.password/>

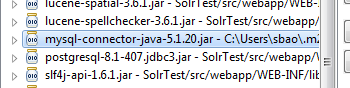
……

</properties>

1. SlorTest/solr/conf/data-config.xml中添加：

<dataSource name=*"JdbcDataSource"* driver=*"com.mysql.jdbc.Driver"* url=*"jdbc:mysql://localhost/LGFAQ"* user=*"root"* password=*""*/>

1. 要吧mysql的jar加到SolrTest的Referenced Lib中；



1. MySQL中把LGFAQ.faq.content的格式换成MediumText，可以储存16M文本内容：

alter table lgfaq.faq modify content MEDIUMTEXT ;

* 可以改变初始数据库内容在“/LGFAQ/src/test/resources/sample-data.xml”
* MySQL Error Code: 1175

You are using safe update mode and you tried to update a table without a WHERE that uses a KEY column

SET SQL\_SAFE\_UPDATES=0;

UPDATE `lgfaq`.`faq` SET time\_edited = null ;

* Packet for query is too large

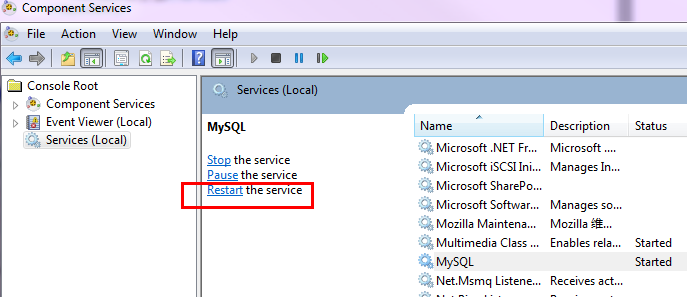
com.mysql.jdbc.PacketTooBigException: Packet for query is too large (1461147 > 1048576). You can change this value on the server by setting the max\_allowed\_packet' variable.

C:\apps\MySQL\MySQL Server 5.5\my.ini

[mysqld]段中的＂max\_allowed\_packet = 1M＂，(可能有些文件会没有这一行的)

将1M更改为4M(如果没有这行内容，增加一行))，保存, 重起MySQL服务。

现在可以load大于1M的文件了



* 控制器中onSubmit方法返回url要带“/”

否则url无法带参数

@RequestMapping(method = RequestMethod.*POST*)

**public** String onSubmit(Faq faq, BindingResult errors, HttpServletRequest request,

HttpServletResponse response)

**throws** Exception {

**return** "redirect:/editfaq?id="+faq.getId();

}

* slf4j+logback vs. log4j+commons-logging

前者组合要远优于后者！

目前是使用后者

* 建立Log4j目录

1. Web.xml中添加：

</context-param>

<context-param>

<param-name>webAppRootKey</param-name>

<param-value>webName.root</param-value>

</context-param>

<!--由Sprng载入的Log4j配置文件位置-->

<context-param>

<param-name>log4jConfigLocation</param-name>

<param-value>/WEB-INF/log4j.properties</param-value>

</context-param>

<!--Spring默认刷新Log4j配置文件的间隔,单位为millisecond-->

<context-param>

<param-name>log4jRefreshInterval</param-name>

<param-value>60000</param-value>

</context-param>

1. 新建文件C:\projects\LGFAQ\src\main\webapp\WEB-INF\log4j.properties，添加：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*log4j.properties\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 设置logger级别 ###

## ERROR、WARN、INFO、DEBUG

log4j.rootLogger=INFO,stdout,A1

### appender.stdout输出到控制台 ###

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.Target=System.out

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern= [%5p] [LGFAQ] %d{yyyy-MM-dd HH:mm:ss}: %-4r [%-5p] [%t] ( %F,%L ) - %m%n

### appender.A1输出到日志文件 ###

log4j.appender.A1=org.apache.log4j.DailyRollingFileAppender

log4j.appender.A1.File=WEB-INF\\logs\\LGFAQ.log

##注意上面日志文件相对应用根目录路径的写法

log4j.appender.A1.DatePattern='.'yyyy-MM-dd'.log'

log4j.appender.A1.Append=true

## 输出DEBUG级别以上的日志

log4j.appender.A1.Threshold=WARN

log4j.appender.A1.layout=org.apache.log4j.PatternLayout

log4j.appender.A1.layout.ConversionPattern= [%5p] [LGFAQ] %d{yyyy-MM-dd HH:mm:ss}: %-4r [%t] ( %F,%L ) - %m%n

1. 新建目录：C:\projects\LGFAQ\src\main\webapp\WEB-INF\logs\

* 彻底关闭HtmlUnit的报错信息

在log4j.properties中添加：

## turn off HtmlUnit page error tracing;

log4j.logger.com=FATAL

log4j.logger.com.gargoylesoftware.htmlunit.javascript=FATAL

* 实体实现单向one to one的思路和步骤

1. Web.xml设置filter

<filter>

<filter-name>lazyLoadingFilter</filter-name>

<filter-class>org.springframework.orm.hibernate3.support.OpenSessionInViewFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>lazyLoadingFilter</filter-name>

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

</filter-mapping>

1. 由于Faq超大的文本内容，在Faq对象载入时会耗费大量系统资源。所以选择把Content属性分离出Faq实体，构成新的实体对象FaqContent，并实现从Faq指向FaqContent的单向一对一懒加载映射。

@Entity

@Table(name = "faq\_content")

**public** **class** FaqContent **extends** BaseObject **implements** Serializable {

**private** **static** **final** **long** *serialVersionUID* = 733653801054360283L;

**private** Long id;

**private** String faqContent;

**public** FaqContent() {

**super**();

**this**.setFaqContent("");

}

@Id

@GeneratedValue(strategy = GenerationType.*AUTO*)

@Column(nullable = **false**)

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

@Column(name = "content")

**public** String getFaqContent() {

**return** faqContent;

}

**public** **void** setFaqContent(String faqContent) {

**this**.faqContent = faqContent;

}

}

1. Faq这边：

@Entity

@Table(name = "faq")

**public** **class** Faq **extends** BaseObject **implements** Serializable {

…

**private** FaqContent faqContent;

…

**public** Faq() {

**…**

**this**.faqContent = **new** FaqContent();

…

}

…

@OneToOne(cascade = { CascadeType.*ALL* }, fetch = FetchType.*LAZY*)

//懒加载：只有当此方法被调用时，才加载FaqContent对象

**public** FaqContent getFaqContent() {

**return** faqContent;

}

**public** **void** setFaqContent(FaqContent faqContent) {

**this**.faqContent = faqContent;

}

…

}

1. 需要在Faq表新建column“faq\_content\_id”,否则有报错”column not found in field list”.
2. 当需要反向调用和双向维护时候，需要一对一的双向映射，要用到mappedby：

对于mappedBy复习下：

a) 只有OneToOne,OneToMany,ManyToMany上才有mappedBy属性，ManyToOne不存在该属性；

b) mappedBy标签一定是定义在the owned side(被拥有方的)，他指向the owning side(拥有方)；

c) mappedBy的含义，应该理解为，拥有方能够自动维护 跟被拥有方的关系；

   当然，如果从被拥有方，通过手工强行来维护拥有方的关系也是可以做到的。

d) mappedBy跟JoinColumn/JoinTable总是处于互斥的一方，可以理解为正是由于拥有方的关联被拥有方的字段存在，拥有方才拥有了被 拥有方。mappedBy这方定义的JoinColumn/JoinTable总是失效的，不会建立对应的字段或者表

人跟身份证双向关联

在Person 里面定义的注解：

@OneToOne(cascade={CascadeType.ALL},optional=true)   
public IDCard getIdCard() {   
         return idCard;   
}

在IDCard 里面定义的注释:   
@OneToOne(cascade=CascadeType.ALL,**mappedBy="idCard"** , optional=false)   
public Person getPerson() {   
       return person;   
}   
多了一个mappedBy这个方法,它表示什么呢?**它表示当前所在表和Person的关系是定义在Person里面的idCard这个成员上面的,它表示此表是一对一关系中的从表,也就是关系是在person表里面维护的 ,这一点很重要** . **Person表是关系的维护者，owner side，有主导权，它有个外键指向IDCard。**我们也可以让主导权在IDCard上面,也就是让它产生一个指向Person的外键,这也是可以的,但是最好是**让Person来维护整个关系,这样更符合我们正常的思维** .

我们也可以看到在Person里面IDCard是注释是optional=true,也就是说一个人是可以没有身份证的,但是一个身份证不可以没有 人,所以在IDCard里面注释person的时候,optional就为false了,这样就可以防止一个空的身份证记录进数据库.

1. Solr这边的更改 “SolrTest/solr/conf/data-config.xml” ：

<document>

<entity name=*"id"* query=*"select id, title, author, link, remark, keyword\_string from faq"* >

<field column=*"id"* name=*"id"*/>

<field column=*"title"* name=*"title"*/>

<field column=*"text"* name=*"text"*/>

<field column=*"author"* name=*"author"*/>

<field column=*"link"* name=*"link"*/>

<field column=*"remark"* name=*"remark"*/>

<field column=*"keywordString"* name=*"keyword\_string"*/>

</entity>

<entity name=*"content"* query=*"select content from faq\_content"* >

<field column=*"content"* name=*"content"*/>

</entity>

</document>

* 在faqs.jsp中的display table 中每一行加入删除本行FAQ的按钮，使用DWR实现删除功能：

<display:table name="allFaqs" class="table" requestURI="/faqs" id="faqTable" >

…

<display:column property="title" sortable="true" href="faqform" media="html" paramId="id" paramProperty="title" titleKey="faq.title"/>

<display:column sortable="false" titleKey="faqlist.operations">

<input type="button" class="editFaqButton button" name="editFaq" onclick="window.location.href='editfaq?id=${faqTable.id}'"

value="<fmt:message key="faqList.edit"/>"/>

<input type="submit" class="delFaqButton button" name="deleteFaq" onclick="delFaqFunction(${faqTable.id}, '${faqTable.title}');"

value="<fmt:message key="faqList.delete"/>" />

</display:column>

…

</display:table>

<script type="text/javascript">

function delFaqFunction(faqId, faqTitle) {

if (confirm('<fmt:message key="confirmBox.deleteFaq"/>\n\n' + 'ID\t\t<fmt:message

key="confirmBox.deleteFaqTitle"/>\n' + faqId + "\t" + faqTitle) == true) {

DwrService.delFaq(faqId);

alert('FAQ: "' + faqTitle + '" <fmt:message key="confirmBox.deleteFaqDone"/>');

window.location.href='faqs';

}

}

</script>

