## 1,html内容换行

在标签中加入style="word-wrap:break-word;” 可以保正英文单词不被切断。

margin-right:15px 属性是使页面内容 离当前标签的边框的距离。

## 2,table合并行

<table border>

<tr>

<td>title1</td>

<td>title2</td>

</tr>

<tr>

<td>content01</td>

<td rowspan="2">合并行</td>

</tr>

<tr>

<td>content02</td>

</tr>

</table>

被合并的行，在tr里应该少一个td

## 3,Spring MVC上传文件

需要jar包：commons-fileupload-1.2.2.jar,commons-io-2.2.jar

前台：

<form method=*"post"* action=*"/springMVC010/hello/upload"* enctype=*"multipart/form-data"*>

用户名:<input type=*"text"* name=*"str"*><br>

file:<input type=*"file"* class=*"multi"* name=*"file"* maxlength=*20* accept=*"txt|pdf|doc|docx"*>

<input type=*"submit"*/>

</form>

注：enctype=*"multipart/form-data" 这个是form表单必须的。*

***如果想一次性上传多个文件，就是在弹出的选择框里可以选择多个文件，需要 在input 里加入以下内容：***

<input type=*"file"* name=*"files"* **multiple=*"true"***/>

后台：

SpringMVC-Servlet.XML

<bean id=*"multipartResolver"*

class=*"org.springframework.web.multipart.commons.CommonsMultipartResolver"* >

<property name=*"defaultEncoding"* value=*"utf-8"*/> <!-- 默认编码 (ISO-8859-1) -->

<property name=*"maxInMemorySize"* value=*"10240"*/> <!-- 最大内存大小 (10240) -->

<property name=*"uploadTempDir"* value=*"/upload/"*/> <!-- 上传后的目录名 (WebUtils#TEMP\_DIR\_CONTEXT\_ATTRIBUTE) -->

<property name=*"maxUploadSize"* value=*"-1"*/> <!-- 最大文件大小，-1为无限止(-1) -->

</bean>

@RequestMapping(value = "/upload")

@ResponseBody//

**public** String uploadFile(MultipartHttpServletRequest request,

HttpServletResponse response,String str) **throws** UnsupportedEncodingException {

//request.setCharacterEncoding("UTF-8");

String basePath = request.getServletContext().getRealPath("/upload");

FileOutputStream fileOutputStream = **null**;

List<MultipartFile> listUploadFile = request.getFiles("file");

**for** (**int** i = 0; i < listUploadFile.size(); i++) {

MultipartFile file = listUploadFile.get(i);

**if** (!file.isEmpty()) {

String fileName = file.getOriginalFilename();

String name = fileName.split("\\.")[0];

String suffix = fileName.substring(fileName.lastIndexOf("."));

//文件名重新命名 oldName + random + time

**double** r = Math.*random*();

String newFileName = name + "\_" + r + "\_" + System.*currentTimeMillis*()

+ suffix;

String filePath = basePath + "\\" + newFileName;

File files = **new** File(filePath);

**if**(files.exists()){

files.delete();

}

**try** {

//files.createNewFile();

fileOutputStream = **new** FileOutputStream(files);

fileOutputStream.write(file.getBytes());

fileOutputStream.flush();

} **catch** (Exception e) {

e.printStackTrace();

}

**if** (fileOutputStream != **null**) {

**try** {

fileOutputStream.close();

} **catch** (IOException ie) {

ie.printStackTrace();

}

}

}

}

**return** "";

}

## 4,jquery 点击事件对象

当同一个事件被绑定到多个事件的时候，需要在点击当前对象的时候触发，但是又不能触发其它对象所对应的同一个事件。

$(“.testClass”).onclick(function(event){

Var $x = $(event.target);//这样子可以得到鼠标当时点击的页面标签对应的对象。

Var aaa = $x.parent().find(“input”);

aaa.dosome();

});

## 5,InputStream处理文件

InputStream input =  **new** InputStream(**new** FileInputStream(File));

InputStreamReader insr = **new** InputStreamReader(input);

BufferedReader br = **new** BufferedReader(insr);

String str = "";

**while** ((str = br.readLine()) != **null**)

{

System.***out***.println(str);

}

将文件装入InputStream,转化为 Reader,最后以一次读一行的方式将数据打印在控制台。

Eg:

通过 apache httpClient包来获取某地天气数据,最后打印控制台。

需要jar:

Client:

<http://mirrors.cnnic.cn/apache//httpcomponents/httpclient/binary/httpcomponents-client-4.4.1-osgi-bin.zip>

core:

<http://mirrors.cnnic.cn/apache//httpcomponents/httpcore/binary/httpcomponents-core-4.4.1-osgi-bin.zip>

Code:

**public** **final** **static** **void** main(String[] args) **throws** Exception {

CloseableHttpClient httpclient = HttpClients.*createDefault*();

**try** {

HttpGet httpget = **new** HttpGet("http://m.weather.com.cn/data/101010100.html");

System.***out***.println("Executing request " + httpget.getURI());

CloseableHttpResponse response = httpclient.execute(httpget);

**try** {

System.***out***.println("----------------------------------------");

System.***out***.println(response.getStatusLine());

//System.out.println( EntityUtils.toString(response.getEntity()));

HttpEntity httpEntry = response.getEntity();

InputStream input = httpEntry.getContent();

InputStreamReader insr = **new** InputStreamReader(input);

BufferedReader br = **new** BufferedReader(insr);

String str = "";

**while** ((str = br.readLine()) != **null**)

{

System.***out***.println(str);

}

// Do not feel like reading the response body

// Call abort on the request object

httpget.abort();

} **finally** {

response.close();

}

} **finally** {

httpclient.close();

}

}

## 6,Jquery获取boolean类型方法

$(“#id”).prop(“checked”);用来获取jquery对象中boolean 类型的值。（true,false）

## 7,jquery非空判断

Var $test = $(“#test”);

If($test.length > 0)

得到一个jquery对象之后，判断对象的长度是否大于1，如果 大于 则不为空。

var test = $form.find("input[name='test']").val();

if(test){

不为空

}

if(!test){

为空

}

## 8,Set处理数据

如果 set 容器不为空，则set.iterator()一定不为空，可以省去一层非空判断。

用while循环iter一般会让代码多一到两行，可以尝试用for来循环。

Eg:

判断循环多用

if (colorSet == null)  
{  
  continue;  
}

可以少写好多大括号，程序可读性强。

将while处理set转为用for来处理，可以省去多余的代码1-2行。Eg:

for (Iterator<SwatchColorEnum> iterator = colorSet.iterator();iterator.hasNext();) {  
  final SwatchColorEnum swatchColorEnum = iterator.next();  
  // Do something  
}

## 9,jquery 得到多个input type=radio

使用jquery 得到多个radio,如果需要将所有的Radio都加上个属性，或者class,不需要进行循环判断。直接当成一个的使用就可以。

Eg:

**var** $allGradients = $div.find("input[name$='optionGrandient']");

$allGradients.prop("disabled",**false**);

## 10,List使用说明

如果一个jar的返回值是list.那么这个list可能是不可以修改的。这个List是个接口。如下示例：当有个API返回类型为List时，不能臆断这个List就是ArrayList、List的实现太多了。

一般API的提供者如果想保护其内部的List，就会给你返回一个不可修改的List

目的就是为了防止你修改

Collections.unmodifiableList(YourList);

。如果想修改这个List,可以使用如下方法：

List<LensGradientTypeEnum> optionGradientList = new ArrayList<LensGradientTypeEnum>(source.getLensGradient());

这个时候的List 就是可以修改的。

## 11，Jquery 判断一个集合里没有另一个集合的哪个元素。

**判断集合**$gradAll 里哪个元素不在数组array里：

**var** array = **new** Array();

$excludeGradient.each(**function**(){

array.push($(**this**).val());

});

**for**(**var** i=0;i<$gradAll.length;i++){

**var** g = $gradAll.eq(i).attr("data");

**if**(array.indexOf(g) == -1){

**return** g;

}

}

## 12 [cron表达式详解](http://www.cnblogs.com/linjiqin/p/3178452.html)

Cron表达式是一个字符串，字符串以5或6个空格隔开，分为6或7个域，每一个域代表一个含义，Cron有如下两种语法格式：

Seconds Minutes Hours DayofMonth Month DayofWeek Year或Seconds Minutes Hours DayofMonth Month DayofWeek

每一个域可出现的字符如下：   
Seconds:可出现", - \* /"四个字符，有效范围为0-59的整数   
Minutes:可出现", - \* /"四个字符，有效范围为0-59的整数   
Hours:可出现", - \* /"四个字符，有效范围为0-23的整数   
DayofMonth:可出现", - \* / ? L W C"八个字符，有效范围为0-31的整数   
Month:可出现", - \* /"四个字符，有效范围为1-12的整数或JAN-DEc   
DayofWeek:可出现", - \* / ? L C #"四个字符，有效范围为1-7的整数或SUN-SAT两个范围。1表示星期天，2表示星期一， 依次类推   
Year:可出现", - \* /"四个字符，有效范围为1970-2099年

每一个域都使用数字，但还可以出现如下特殊字符，它们的含义是：   
(1)\*：表示匹配该域的任意值，假如在Minutes域使用\*, 即表示每分钟都会触发事件。

(2)?:只能用在DayofMonth和DayofWeek两个域。它也匹配域的任意值，但实际不会。因为DayofMonth和DayofWeek会相互影响。例如想在每月的20日触发调度，不管20日到底是星期几，则只能使用如下写法： 13 13 15 20 \* ?, 其中最后一位只能用？，而不能使用\*，如果使用\*表示不管星期几都会触发，实际上并不是这样。   
  
(3)-:表示范围，例如在Minutes域使用5-20，表示从5分到20分钟每分钟触发一次   
  
(4)/：表示起始时间开始触发，然后每隔固定时间触发一次，例如在Minutes域使用5/20,则意味着5分钟触发一次，而25，45等分别触发一次.   
  
(5),:表示列出枚举值值。例如：在Minutes域使用5,20，则意味着在5和20分每分钟触发一次。   
  
(6)L:表示最后，只能出现在DayofWeek和DayofMonth域，如果在DayofWeek域使用5L,意味着在最后的一个星期四触发。   
  
(7)W:表示有效工作日(周一到周五),只能出现在DayofMonth域，系统将在离指定日期的最近的有效工作日触发事件。例如：在 DayofMonth使用5W，如果5日是星期六，则将在最近的工作日：星期五，即4日触发。如果5日是星期天，则在6日(周一)触发；如果5日在星期一到星期五中的一天，则就在5日触发。另外一点，W的最近寻找不会跨过月份   
  
(8)LW:这两个字符可以连用，表示在某个月最后一个工作日，即最后一个星期五。   
  
(9)#:用于确定每个月第几个星期几，只能出现在DayofMonth域。例如在4#2，表示某月的第二个星期三。

举几个例子:   
0 0 2 1 \* ? \* 表示在每月的1日的凌晨2点调度任务   
0 15 10 ? \* MON-FRI 表示周一到周五每天上午10：15执行作业   
0 15 10 ? 6L 2002-2006 表示2002-2006年的每个月的最后一个星期五上午10:15执行作

一个cron表达式有至少6个（也可能7个）有空格分隔的时间元素。   
按顺序依次为   
秒（0~59）   
分钟（0~59）   
小时（0~23）   
天（月）（0~31，但是你需要考虑你月的天数）   
月（0~11）   
天（星期）（1~7 1=SUN 或 SUN，MON，TUE，WED，THU，FRI，SAT）   
年份（1970－2099）

其中每个元素可以是一个值(如6),一个连续区间(9-12),一个间隔时间(8-18/4)(/表示每隔4小时),一个列表(1,3,5),通配符。由于"月份中的日期"和"星期中的日期"这两个元素互斥的,必须要对其中一个设置?

0 0 10,14,16 \* \* ? 每天上午10点，下午2点，4点   
0 0/30 9-17 \* \* ? 朝九晚五工作时间内每半小时   
0 0 12 ? \* WED 表示每个星期三中午12点   
"0 0 12 \* \* ?" 每天中午12点触发   
"0 15 10 ? \* \*" 每天上午10:15触发   
"0 15 10 \* \* ?" 每天上午10:15触发   
"0 15 10 \* \* ? \*" 每天上午10:15触发   
"0 15 10 \* \* ? 2005" 2005年的每天上午10:15触发   
"0 \* 14 \* \* ?" 在每天下午2点到下午2:59期间的每1分钟触发   
"0 0/5 14 \* \* ?" 在每天下午2点到下午2:55期间的每5分钟触发   
"0 0/5 14,18 \* \* ?" 在每天下午2点到2:55期间和下午6点到6:55期间的每5分钟触发   
"0 0-5 14 \* \* ?" 在每天下午2点到下午2:05期间的每1分钟触发   
"0 10,44 14 ? 3 WED" 每年三月的星期三的下午2:10和2:44触发   
"0 15 10 ? \* MON-FRI" 周一至周五的上午10:15触发   
"0 15 10 15 \* ?" 每月15日上午10:15触发   
"0 15 10 L \* ?" 每月最后一日的上午10:15触发   
"0 15 10 ? \* 6L" 每月的最后一个星期五上午10:15触发   
"0 15 10 ? \* 6L 2002-2005" 2002年至2005年的每月的最后一个星期五上午10:15触发   
"0 15 10 ? \* 6#3" 每月的第三个星期五上午10:15触发

有些子表达式能包含一些范围或列表

例如：子表达式（天（星期））可以为 “MON-FRI”，“MON，WED，FRI”，“MON-WED,SAT”

“\*”字符代表所有可能的值

因此，“\*”在子表达式（月）里表示每个月的含义，“\*”在子表达式（天（星期））表示星期的每一天

“/”字符用来指定数值的增量   
例如：在子表达式（分钟）里的“0/15”表示从第0分钟开始，每15分钟   
在子表达式（分钟）里的“3/20”表示从第3分钟开始，每20分钟（它和“3，23，43”）的含义一样

“？”字符仅被用于天（月）和天（星期）两个子表达式，表示不指定值   
当2个子表达式其中之一被指定了值以后，为了避免冲突，需要将另一个子表达式的值设为“？”

“L” 字符仅被用于天（月）和天（星期）两个子表达式，它是单词“last”的缩写   
但是它在两个子表达式里的含义是不同的。   
在天（月）子表达式中，“L”表示一个月的最后一天   
在天（星期）自表达式中，“L”表示一个星期的最后一天，也就是SAT

如果在“L”前有具体的内容，它就具有其他的含义了

例如：“6L”表示这个月的倒数第６天，“FRIL”表示这个月的最一个星期五   
注意：在使用“L”参数时，不要指定列表或范围，因为这会导致问题

字段 允许值 允许的特殊字符   
秒 0-59 , - \* /   
分 0-59 , - \* /   
小时 0-23 , - \* /   
日期 1-31 , - \* ? / L W C   
月份 1-12 或者 JAN-DEC , - \* /   
星期 1-7 或者 SUN-SAT , - \* ? / L C #   
年（可选） 留空, 1970-2099 , - \* /

## 13，Spring执行定时任务

通过maven项目来实现，pom文件中加入spring引用：

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>3.2.14.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework.webflow</groupId>

<artifactId>spring-webflow</artifactId>

<version>2.3.4.RELEASE</version>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.1.0</version>

</dependency>

日志记录使用logback.xml:

<configuration>

<appender name=*"FILE"* class=*"ch.qos.logback.core.rolling.RollingFileAppender"*>

<file>${catalina.home}/log/schedule.log</file>

<rollingPolicy class=*"ch.qos.logback.core.rolling.TimeBasedRollingPolicy"*>

<fileNamePattern>${catalina.home}/log/schedule-%d{yyyy-MM-dd}.log</fileNamePattern>

</rollingPolicy>

- <encoder>

<Pattern>%d [%-5level][%t][%c][%X{tenant}][%X{requestId}] %m%n</Pattern>

</encoder>

</appender>

<appender name=*"STDOUT"* class=*"ch.qos.logback.core.ConsoleAppender"*>

<encoder>

<Pattern>%d [%-5level][%t][%c][%X{tenant}][%X{requestId}] %m%n</Pattern>

</encoder>

</appender>

<logger name=*"javax.validation"* level=*"info"* />

<logger name=*"org.springframework.beans.factory"* level=*"warn"*/>

<logger name=*"org.springframework.transaction"* level=*"info"* />

<logger name=*"org.springframework.validation"* level=*"info"* />

<logger name=*"org.springframework.test"* level=*"warn"* />

<logger name=*"org.springframework.aop.interceptor.PerformanceMonitorInterceptor"* level=*"trace"* appender-ref=*"STDOUT"*/>

<!-- DB statements -->

<logger name=*"com.hybris.kernel.engine.jdbc.JDBCPersistenceEngine"* level=*"info"* />

<logger name=*"org.springframework.jdbc.core.JdbcTemplate"* level=*"info"* />

<root level=*"info"*>

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

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

</root>

</configuration>

Web.xml设置spring监听：

<context-param>

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

<param-value>classpath\*:spring/spring.xml</param-value>

</context-param>

<listener>

<description>Spring Context</description>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

Spring.xml配置定时任务：

<context:component-scan base-package=*"com.nicchagil.\*"*/>

<!-- <task:annotation-driven/> -->

<task:annotation-driven scheduler=*"myScheduler"*/>

<task:scheduler id=*"myScheduler"* pool-size=*"20"*/>

相关java代码：

package com.nicchagil.springtask;

import java.util.concurrent.TimeUnit;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.scheduling.annotation.Scheduled;

import org.springframework.stereotype.Component;

@Component

public class MyFirstSpringJob {

private final Logger logger = LoggerFactory.getLogger(this.getClass());

//每隔10秒开始执行

@Scheduled(cron = "0/10 \* \* \* \* ?")

public void run() {

logger.info("MyFirstSpringJob trigger...");

/\* 模拟此Job需耗时5秒 \*/

try {

TimeUnit.SECONDS.sleep(5);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

package com.nicchagil.springtask;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.scheduling.annotation.Scheduled;

import org.springframework.stereotype.Component;

@Component

public class MySecondSpringJob {

private final Logger logger = LoggerFactory.getLogger(this.getClass());

//任意时间的第3秒执行

@Scheduled(cron = "3 \* \* \* \* ?")

public void run() {

logger.info("MySecondSpringJob trigger...");

}

}

只需以上配置，就可以进行spring管理 的任务。定时是通过 cron时间管理。

## 14 解析银联返回字符串

第三方返回字符串如下 ，使用java自带的Spring.splt(“\n”)在程序运行时无法对字符进行分隔。但是通过**org.apache.commons.lang.StringUtils的split()**方法可以对其进行分解：

String response = “20160324\n2.00\nAMY160380022006\n99800568\n978033978033\n00\n0324\n&UPOPPayModeCode=LitePay&UPOPPayModeName=????\n\n2016-03-24 15:45:42.887\n2016-03-24 15:45:57.0\n6216261000000000018\n”

String[] queryArray = StringUtils.split(response, “\\n”);

如果\n之间没有数据，是空的，会导致分隔时当成一个字符。

## 15 Hybris retry功能

指定可以执行重复次数的功能：

**final** TaskModel task = (TaskModel) modelService.create(TaskModel.**class**);

task.setRetry(0);

task.setRunnerBean("lynxPushtoCustomsTaskRunner");

**final** TaskConditionModel tm = (TaskConditionModel) modelService.create(TaskConditionModel.**class**);

tm.setUniqueID(mapData.get("orderId"));

task.setConditions(Collections.*singleton*(tm));

task.setContext(mapData);

taskService.scheduleTask(task);

taskService.triggerEvent(mapData.get("orderId"));

1. 定义一个TaskModel,为这个taskModel设置retry的次数，以及这个task执行的Runner。
2. 定义TaskConditionModel 设置他执行的条件及关联关系。
3. 装condition设置到task中。
4. 调用de.hybris.platform.task.taskService,将任务设置进去。
5. 通过taskService触发任务。

注：触发的event 是在 condition中设置的 uniqueId。

Runner类是实现TaskRunner的一个bean:

public class LynxPushtoCustomsTaskRunner implements TaskRunner<TaskModel>{

要执行的方法内容写在 :

@Override

public void run(final TaskService paramTaskService, final TaskModel task)

{

//当前重试次数与最大重试次数，确定是否继续执行

// maxRetries可在spring中进行配置

if (task.getRetry() < maxRetries){

}}

2. retry demo

通过页面去调用controller方法,controller方法会解发一个事件去执行retry .

1.controller方法

|  |
| --- |
| **import** de.hybris.platform.servicelayer.model.ModelService;  **import** de.hybris.platform.task.TaskConditionModel;  **import** de.hybris.platform.task.TaskModel;  **import** de.hybris.platform.task.TaskService;  @RequestMapping(value = "/test02", method =  { RequestMethod.***GET***, RequestMethod.***POST*** })  @ResponseBody  **public** String test02(**final** HttpServletRequest request)  {  ***LOG***.info("====test retry start=====");  // create models  **final** TaskModel task = modelService.create(TaskModel.**class**);  **final** TaskConditionModel cond = modelService.create(TaskConditionModel.**class**);  // configure them  task.setRunnerBean("testRetryFunctionBean");  // define event name  cond.setUniqueID("testRetryFunction");  // add to task  task.setConditions(Collections.*singleton*(cond));  task.setContext("hello");  // schedule  taskService.scheduleTask(task);  taskService.triggerEvent("testRetryFunction");  ***LOG***.info("====test retry end=====");  **return** "hello world";  } |

2.相关Runnerbean定义

|  |
| --- |
| **package** com.npz.icolor.icolorpaymentaddon;  **import** de.hybris.platform.task.RetryLaterException;  **import** de.hybris.platform.task.TaskModel;  **import** de.hybris.platform.task.TaskRunner;  **import** de.hybris.platform.task.TaskService;  **import** org.apache.log4j.Logger;  **public** **class** TestRetryFunctionBean **implements** TaskRunner<TaskModel>  {  **private** **static** **final** Logger ***LOG*** = Logger.*getLogger*(TestRetryFunctionBean.**class**);  @Override  **public** **void** handleError(**final** TaskService arg0, **final** TaskModel arg1, **final** Throwable arg2)  {  ***LOG***.info("===========handleError=========");  }  @Override  **public** **void** run(**final** TaskService task, **final** TaskModel taskModel) **throws** RetryLaterException  {  **if** (taskModel != **null**)  {  **final** String context = (String) taskModel.getContext();  ***LOG***.info("======" + context + "=====run=========");  }  **else**  {  ***LOG***.error("=========run error======");  }  }  } |

3.RunnerBean注入

|  |
| --- |
| <bean id=*"testRetryFunctionBean"* class=*"com.npz.icolor.icolorpaymentaddon.TestRetryFunctionBean"*></bean> |

## 16 Hybris cronjob

* 定义cronjob。在core-item.xml中定义job的Model.

<itemtype code=*"LynxCheckPush2CustomsJob"* extends=*"CronJob"* autocreate=*"true"* generate=*"true"* jaloclass=*"com.amway.lynx.core.jalo.LynxCheckPush2CustomsJob"*><description>CronJob send push to customs not success email.</description></itemtype>

* 定义cronjob工作类。

**public** **class** LynxCheckPush2CustomsJobPerformable **extends** AbstractJobPerformable<LynxCheckPush2CustomsJobModel>

{

**private** **static** **final** Logger ***LOG*** = Logger.*getLogger*(LynxCheckPush2CustomsJobPerformable.**class**.getName());

**private** LynxOrderPaymentService lynxOrderPaymentService;

@Override

**public** PerformResult perform(**final** LynxCheckPush2CustomsJobModel cronJob)

{

**if** (***LOG***.isDebugEnabled())

{

***LOG***.debug("start check order exported Job Running...");

}

**final** List<OrderModel> orderList = lynxOrderPaymentService.getPush2CustomsNOTsuccessOrder(DeviceType.***DESKTOP***,

PaymentStatus.***PAID***, MobilePush2CustomsStatus.***SUCCESS***);

**if** (!CollectionUtils.*isEmpty*(orderList))

{

lynxOrderPaymentService.sendEmail(orderList);

}

**else**

{

**if** (***LOG***.isDebugEnabled())

{

***LOG***.debug("there is no mobile order push to customs failure,need not send email");

}

}

**return** **new** PerformResult(CronJobResult.***SUCCESS***, CronJobStatus.***FINISHED***);

}

注意：AbstractJobPerformable<LynxCheckPush2CustomsJobModel> 和perform(**final** LynxCheckPush2CustomsJobModel cronJob) 里面的参数 ：LynxCheckPush2CustomsJobModel 一定要是刚刚在core-item.xml里定义的Model.

将Cronjob注入到spring.中。

* impex中进行配置cronjob.

/lynxinitialdata/resources/lynxinitialdata/import /coredata/common/essential-data.impex

一般在这个文件中进行Cronjob的配置：

1，定义cronjob

INSERT\_UPDATE LynxCheckPush2CustomsJob;code[unique=true];job(code);sessionLanguage(isoCode)[default=zh];sessionCurrency(isoCode)[default=CNY]

;lynxCheckPush2CustomsJob;lynxCheckPush2CustomsJobPerformable

2,定义触发Cronjob的trigger(触发的时间)

INSERT\_UPDATE Trigger;cronJob(code)[unique=true];cronExpression

;lynxResetAggOrderByIdNumberEveryYearCronJob;0 0 0 1 1 ?

;lynxCheckPush2CustomsJob; 0 0/5 \* \* \* ?

## 17 Hybris 邮件

Init:

重写hybris邮件API:

<alias alias=*"emailGenerationService"* name=*"lynxDefaultEmailGenerationService"*/>

<bean id=*"lynxDefaultEmailGenerationService"*

class=*"com.amway.lynx.core.dataexport.impl.LynxDefaultEmailGenerationService"*

parent=*"abstractBusinessService"*>

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

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

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

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

</bean>

**public** **class** LynxDefaultEmailGenerationService **extends** DefaultEmailGenerationService {

**private** **static** **final** String ***CHECK\_ORDER\_EXPORTED\_CC\_EMAIL*** = "mail.smtp.contact.us.cc.email";

**private** ConfigurationService configurationService;

**protected** EmailMessageModel createEmailMessage(**final** String emailSubject, **final** String emailBody,

**final** AbstractEmailContext<BusinessProcessModel> emailContext)

{

**final** List<EmailAddressModel> toEmails = **new** ArrayList<EmailAddressModel>();

String toEmailString = emailContext.getToEmail();

String toDisplayName = "";

setEmailAddress(toEmailString,toDisplayName,toEmails);

**final** String ccEmailsAddress = configurationService.getConfiguration().getString(***CHECK\_ORDER\_EXPORTED\_CC\_EMAIL***);

**final** List<EmailAddressModel> ccEmails = **new** ArrayList<EmailAddressModel>();

String ccDisplayName = "";

setEmailAddress(ccEmailsAddress,ccDisplayName,ccEmails);

**final** EmailAddressModel fromAddress = getEmailService().getOrCreateEmailAddressForEmail(emailContext.getFromEmail(), emailContext.getFromDisplayName());

**return** getEmailService().createEmailMessage(toEmails, ccEmails,

**new** ArrayList<EmailAddressModel>(), fromAddress, emailContext.getFromEmail(), emailSubject, emailBody, **null**);

}

**private** **void** setEmailAddress(String emailAdds , String displayName , List<EmailAddressModel> emailLists){

**if**(StringUtils.*isNotBlank*(emailAdds)){

**if** (emailAdds.contains(";")) {

**final** String[] emails = StringUtils.*split*(emailAdds, ";");

**for** (String email : emails) {

**if**(StringUtil.*isNotEmpty*(email)){

**final** EmailAddressModel ccAddress = getEmailService().getOrCreateEmailAddressForEmail(email,

displayName);

emailLists.add(ccAddress);

}

}

} **else** {

**final** EmailAddressModel ccAddress = getEmailService().getOrCreateEmailAddressForEmail(emailAdds,displayName);

emailLists.add(ccAddress);

}

}

}

**public** ConfigurationService getConfigurationService() {

**return** configurationService;

}

**public** **void** setConfigurationService(ConfigurationService configurationService) {

**this**.configurationService = configurationService;

}

<bean id=*"abstractGenerateEmailAction"* class=*"de.hybris.platform.acceleratorservices.process.email.actions.GenerateEmailAction"*

parent=*"abstractAction"* abstract=*"true"*>

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

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

<property name=*"contextResolutionStrategy"* ref=*"processContextResolutionStrategy"*/>

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

</bean>

1. core-spring.xml中定义邮件引用的模版：

<bean id="generateCheckPush2CustomsEmail" parent="abstractGenerateEmailAction">

<property name="frontendTemplateName" value="CheckPush2CustomsEmailTemplate"/>

</bean>

1. 定义发邮件的processModel

<itemtype code="LynxCheckPush2CustomsProcess" autocreate="true"

generate="true" jaloclass="de.hybris.platform.ordersplitting.jalo.LynxCheckPush2CustomsProcess"

extends="OrderProcess">

<attributes>自定义属性（在邮件里会展示出来的属性）

<attribute type="orderNumberList" qualifier="orderNumbers">

<persistence type="property"></persistence>

</attribute>

</attributes>

</itemtype>

1. 在core-spring.xml中定义发送邮件的process:

/lynxcore/resources/lynxcore/processes/sendCheckPush2CustomsEmailProcess.xml

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

<process xmlns=*"http://www.hybris.de/xsd/processdefinition"* start=*"generateCheckPush2CustomsEmail"* name=*"sendCheckPush2CustomsEmailProcess"*

processClass=*"de.hybris.platform.ordersplitting.model.LynxCheckPush2CustomsProcessModel"* onError=*"error"*>

<action id=*"generateCheckPush2CustomsEmail"* bean=*"generateCheckPush2CustomsEmail"*>

<transition name=*"OK"* to=*"sendEmail"*/>

<transition name=*"NOK"* to=*"error"*/>

</action>

<action id=*"sendEmail"* bean=*"sendEmail"*>

<transition name=*"OK"* to=*"removeSentEmail"*/>

<transition name=*"NOK"* to=*"failed"*/>

</action>

<action id=*"removeSentEmail"* bean=*"removeSentEmail"*>

<transition name=*"OK"* to=*"success"*/>

<transition name=*"NOK"* to=*"error"*/>

</action>

<end id=*"error"* state=*"ERROR"*>Something went wrong.</end>

<end id=*"failed"* state=*"FAILED"*>Could not send Check Push to Customs email.</end>

<end id=*"success"* state=*"SUCCEEDED"*>Sent Check Push to Customs email.</end>

</process>

4，core-spring.xml注入process:

<bean id="*sendCheckPush2CustomsEmailProcess*"

class="de.hybris.platform.processengine.definition.ProcessDefinitionResource">

<property name="resource" value="classpath:/lynxcore/processes/*sendCheckPush2CustomsEmailProcess*.xml" />

</bean>

5,调用email API发送邮件。启动process：

@Transactional

**public** **void** sendEmail(**final** List<OrderModel> orders)

{

**final** LynxCheckPush2CustomsProcessModel orderProcessModel = (LynxCheckPush2CustomsProcessModel) getBusinessProcessService()

.createProcess("sendCheckPush2CustomsEmailProcess-" + System.*currentTimeMillis*(),

"sendCheckPush2CustomsEmailProcess");

**final** List<String> orderNumbers = **new** ArrayList<String>();

**if** (!CollectionUtils.*isEmpty*(orders))

{

**for** (**final** OrderModel order : orders)

{

**final** String orderNumber = order.getCode();

orderNumbers.add(orderNumber);

}

orderProcessModel.setOrder(orders.get(0));//order里有 site 信息，发邮件时需要，所以setOrder.

}

orderProcessModel.setOrderNumbers(orderNumbers);//自定义属性

getModelService().save(orderProcessModel);//一定要Save,然后startProcess

getBusinessProcessService().startProcess(orderProcessModel);

}

1. 定义邮件CMS模板：

文件位置：

/lynxinitialdata/resources/lynxinitialdata/import/coredata/contentCatalogs/lynxContentCatalog/email-content.impex

# Import the CMS content for the Lynx site emails

$contentCatalog=lynxContentCatalog

$contentCV=catalogVersion(CatalogVersion.catalog(Catalog.id[default=$contentCatalog]),CatalogVersion.version[default=Online])[default=$contentCatalog:Online]

# Import modulegen config properties into impex macros

UPDATE GenericItem[processor=de.hybris.platform.commerceservices.impex.impl.ConfigPropertyImportProcessor];pk[unique=true]

$jarResourceCms=$config-jarResourceCmsValue

$wideContent=CMSImageComponent,BannerComponent

$emailPackageName=$config-emailContextPackageName

# Email page Template

INSERT\_UPDATE EmailPageTemplate ;$contentCV[unique=true] ;uid[unique=true] ;name ;active ;frontendTemplateName ;subject(code) ;htmlTemplate(code) ;restrictedPageTypes(code)

; ;CheckPush2CustomsEmailTemplate ;Check Push to Customs Email Template ;true ;CheckPush2CustomsEmail ;lynx\_Email\_Check\_Push\_To\_Customs\_Subject ;lynx\_Email\_Check\_Push\_To\_Customs\_Body ;EmailPage

# Templates for CMS Cockpit Page Edit

UPDATE EmailPageTemplate;$contentCV[unique=true];uid[unique=true];velocityTemplate[translator=de.hybris.platform.commerceservices.impex.impl.FileLoaderValueTranslator]

;;CheckPush2CustomsEmailTemplate;$jarResourceCms/structure-view/structure\_CheckPush2CustomsEmailTemplate.vm

INSERT\_UPDATE ContentSlotName;name[unique=true];template(uid,$contentCV)[unique=true][default='CheckPush2CustomsEmailTemplate'];validComponentTypes(code)

;SiteLogo;;;logo

;TopContent;;$wideContent;

;BottomContent;;$wideContent;

INSERT\_UPDATE ContentSlotForTemplate;$contentCV[unique=true];uid[unique=true];position[unique=true];pageTemplate(uid,$contentCV)[unique=true][default='CheckPush2CustomsEmailTemplate'];contentSlot(uid,$contentCV)[unique=true];allowOverwrite

;;SiteLogo-CheckPush2CustomsEmail;SiteLogo;;EmailSiteLogoSlot;true

;;TopContent-CheckPush2CustomsEmail;TopContent;;EmailTopSlot;true

;;BottomContent-CheckPush2CustomsEmail;BottomContent;;EmailBottomSlot;true

# Email Pages

INSERT\_UPDATE EmailPage;$contentCV[unique=true];uid[unique=true];name;masterTemplate(uid,$contentCV);defaultPage;approvalStatus(code)[default='approved']

;;CheckPush2CustomsEmail;Check Order push to customs Email;CheckPush2CustomsEmailTemplate;true;

# Email velocity templates

INSERT\_UPDATE RendererTemplate;code[unique=true];contextClass;rendererType(code)[default='velocity']

;lynx\_Email\_Check\_Push\_To\_Customs\_Subject;$emailPackageName.CheckPush2CustomsEmailContext

;lynx\_Email\_Check\_Push\_To\_Customs\_Body;$emailPackageName.CheckPush2CustomsEmailContext

UPDATE EmailPage;$contentCV[unique=true];uid[unique=true];previewImage(code, $contentCV)

;;CheckPush2CustomsEmail;EmailPageModel\_preview

/lynxinitialdata/resources/lynxinitialdata/import/coredata/contentCatalogs/lynxContentCatalog/email-content\_zh.impex

# Import the CMS content for the Lynx site emails

$contentCatalog=lynxContentCatalog

$contentCV=catalogVersion(CatalogVersion.catalog(Catalog.id[default=$contentCatalog]),CatalogVersion.version[default=Online])[default=$contentCatalog:Online]

# Import config properties into impex macros for modulegen

UPDATE GenericItem[processor=de.hybris.platform.commerceservices.impex.impl.ConfigPropertyImportProcessor];pk[unique=true]

$emailResource=$config-emailResourceValue

# Language

$lang=zh

# Email Pages

UPDATE EmailPage;$contentCV[unique=true];uid[unique=true];fromEmail[lang=$lang];fromName[lang=$lang]

;;CheckPush2CustomsEmail;"lynx@amway.com";"客服团队"

# CMS components and Email velocity templates

UPDATE RendererTemplate;code[unique=true];description[lang=$lang];templateScript[lang=$lang,translator=de.hybris.platform.commerceservices.impex.impl.FileLoaderValueTranslator]

;lynx\_Email\_Check\_Push\_To\_Customs\_Body;"Check Push To Customs Email Body";$emailResource/email-CheckPushToCustomsBody.vm

;lynx\_Email\_Check\_Push\_To\_Customs\_Subject;"Check Push To Customs Email Subject";$emailResource/email-CheckPushToCustomsSubject.vm

/lynxinitialdata/resources/lynxinitialdata/import/coredata/contentCatalogs/lynxContentCatalog/email-content\_en.impex

# Import the CMS content for the Lynx site emails

$contentCatalog=lynxContentCatalog

$contentCV=catalogVersion(CatalogVersion.catalog(Catalog.id[default=$contentCatalog]),CatalogVersion.version[default=Online])[default=$contentCatalog:Online]

# Import config properties into impex macros for modulegen

UPDATE GenericItem[processor=de.hybris.platform.commerceservices.impex.impl.ConfigPropertyImportProcessor];pk[unique=true]

$emailResource=$config-emailResourceValue

# Language

$lang=en

# Email Pages

UPDATE EmailPage;$contentCV[unique=true];uid[unique=true];fromEmail[lang=$lang];fromName[lang=$lang]

;;CheckPush2CustomsEmail;"lynx@amway.com";"Customer Services"

# CMS components and Email velocity templates

UPDATE RendererTemplate;code[unique=true];description[lang=$lang];templateScript[lang=$lang,translator=de.hybris.platform.commerceservices.impex.impl.FileLoaderValueTranslator]

;lynx\_Email\_Check\_Push\_To\_Customs\_Body;"Check Push To Customs Email Body";$emailResource/email-CheckPushToCustomsBody.vm

;lynx\_Email\_Check\_Push\_To\_Customs\_Subject;"Check Push To Customs Email Subject";$emailResource/email-CheckPushToCustomsSubject.vm

1. 邮件页面调用自定义属性：

public class CheckPush2CustomsEmailContext extends AbstractEmailContext<LynxCheckPush2CustomsProcessModel>

{

private static final Logger LOG = Logger.getLogger(CheckPush2CustomsEmailContext.class);

private static final String CHECK\_ORDER\_EXPORTED\_ADMIN\_EMAIL = "mail.smtp.contact.us.to.email";

private Converter<OrderModel, OrderData> orderConverter;

private ConfigurationService configurationService;

public static final String ORDER\_NUMBERS = "orderNumbers";

@Override

public void init(final LynxCheckPush2CustomsProcessModel orderProcessModel, final EmailPageModel emailPageModel)

{

super.init(orderProcessModel, emailPageModel);

final BaseSiteModel baseSite = getSite(orderProcessModel);

final SiteBaseUrlResolutionService siteBaseUrlResolutionService = getSiteBaseUrlResolutionService();

if (baseSite == null)

{

LOG.error("Failed to lookup Site for BusinessProcess [" + orderProcessModel + "]");

}

else

{

put(BASE\_SITE, baseSite);

setUrlEncodingAttributes(getUrlEncoderService().getUrlEncodingPatternForEmail(orderProcessModel));

// Lookup the site specific URLs

put(BASE\_URL, siteBaseUrlResolutionService.getWebsiteUrlForSite(baseSite, getUrlLanguageAttributes(), false, ""));

put(BASE\_THEME\_URL, siteBaseUrlResolutionService.getWebsiteUrlForSite(baseSite, false, ""));

put(SECURE\_BASE\_URL, siteBaseUrlResolutionService.getWebsiteUrlForSite(baseSite, getUrlLanguageAttributes(), true, ""));

put(MEDIA\_BASE\_URL, siteBaseUrlResolutionService.getMediaUrlForSite(baseSite, false));

put(MEDIA\_SECURE\_BASE\_URL, siteBaseUrlResolutionService.getMediaUrlForSite(baseSite, true));

put(THEME, baseSite.getTheme() != null ? baseSite.getTheme().getCode() : null);

}

final String emails = configurationService.getConfiguration().getString(CHECK\_ORDER\_EXPORTED\_ADMIN\_EMAIL);

final List<String> orderNumbers = orderProcessModel.getOrderNumbers();

put(EMAIL, emails);

put(FROM\_EMAIL, emailPageModel.getFromEmail());

put(ORDER\_NUMBERS, orderNumbers);//自定义属性

}

1. email内容：

email-CheckUnionpayCallbackDataBody.vm

需要定义自己邮件的国际化properties文件

## messageSource=classpath:/lynxcore/messages/email-checkOrderPush2Customs\_$lang.properties

#macro(genHtmlLinkStartTag $url)

<a href="$url"><font color="#666666">

#end

#macro(genHtmlLinkEndTag)

</font></a>

#end

#macro(genHtmlLink $url $textColor $bodyContent)

<a href="$url"><font color="$textColor">$bodyContent</font></a>

#end

<html>

<head>

</head>

<body bgcolor="#ffffff"

<table width="100%" border="0" align="center" cellpadding="0" cellspacing="0" bgcolor="#ffffff">

<tr>

<td align="center" valign="top">

<table width="750" border="0" align="center" cellpadding="0" cellspacing="0" bordercolor="#fff">

<tr>

<table width="650" cellpadding="0" cellspacing="0" border="0" align="center">

<tr>

<td valign="middle">&nbsp;</td>

</tr>

<tr>

<td align="left" valign="top">

<p><font color="#666666" size="2" face="Arial, Helvetica, sans-serif"><b>${ctx.messages.salutation}</b></font></p>

<p>

<font color="#666666" size="2" face="Arial, Helvetica, sans-serif">

<b>

#if($ctx.orderNumbers.size() > 0)

${ctx.messages.push2customs\_order\_title}

<br>

<table style="border-collapse:collapse;border-spacing:0;border-left:1px solid #888;border-top:1px solid #888;background:#efefef;">

#foreach($number in $ctx.orderNumbers)

<tr bgcolor="#FFFFFF">

<td style="border-right:1px solid #888;border-bottom:1px solid #888;padding:5px 15px;">

$number

</td>

</tr>

#end

</table>

#end

</b>

</font>

</p>

</td>

</tr>

<tr>

<td align="right" valign="middle">

<p><font color="#666666" size="2" face="Arial, Helvetica, sans-serif"><b>${ctx.messages.signature}</b></font></p>

</td>

</tr>

</table>

</tr>

</table>

</td>

</tr>

</table>

</body>

</html>

1. 邮件local.properties配置：

###########SEND EMAIL CONFIG###########

emailservice.send.enabled=true

mail.smtp.auth=true

mail.smtp.server=10.32.148.123

mail.smtp.port=25

#mail.smtp.user=

#mail.smtp.password=

# set the admin email address

mail.smtp.contact.us.to.email=aaa@objectivasoftware.com;bbb@qq.com

mail.smtp.contact.us.cc.email=ccc@objectivasoftware.com;ddd@objectivasoftware.com

###########SEND EMAIL CONFIG###########

## 18使用solr进行PLP数据展示

Solr展示数据的原理是，solr通过sql把需要展示的数据先从DB里查询出来，保存到solr自己的数据库中，然后页面展示数据时，再从solr中获取。Solr每隔一定的时间会自动执行一次，去产品数据库中更新产品，也就是solr同步。

项目中需要在impex中进行solr的配置：

Impex位置：

/lynxinitialdata/resources/lynxinitialdata/import/coredata/stores/lynx/solr.impex

### 配置solrServer:

$serverConfigName=lynxSolrServerConfig

# Create the solr server configuration

INSERT\_UPDATE SolrServerConfig;name[unique=true];mode(code);embeddedMaster

;$serverConfigName;standalone;false;

### 配置solr分页数据量

# Create the faceted search configuration

INSERT\_UPDATE SolrSearchConfig;description[unique=true];pageSize

;$searchConfigName;20

### 定义solr排序字段：

# Define the available sorts

INSERT\_UPDATE SolrSort;&sortRefID;indexedType(identifier)[unique=true];code[unique=true];useBoost

;sortRef1;$solrIndexedType;relevance;true

;sortRef3;$solrIndexedType;name-asc;false

;sortRef4;$solrIndexedType;name-desc;false

;sortRef5;$solrIndexedType;price-asc;false

;sortRef6;$solrIndexedType;price-desc;false

# Define the sort fields

INSERT\_UPDATE SolrSortField;sort(indexedType(identifier),code)[unique=true] ;fieldName[unique=true];ascending[unique=true]

;$solrIndexedType:relevance ;inStockFlag;false

;$solrIndexedType:relevance ;score;false

;$solrIndexedType:name-asc ;name;true

;$solrIndexedType:name-desc ;name;false

;$solrIndexedType:price-asc ;basePrice;true

;$solrIndexedType:price-desc ;basePrice;false

### 申明排序方式：

# Declare the indexed type Product

INSERT\_UPDATE SolrIndexedType;identifier[unique=true];type(code);variant;sorts(&sortRefID)

;$solrIndexedType;Product;false;sortRef1,sortRef3,sortRef4,sortRef5,sortRef6

### 定义solr查询方式：

$productCatalog=lynxProductCatalog

$catalogVersions=catalogVersions(catalog(id),version);

$serverConfigName=lynxSolrServerConfig

$indexConfigName=lynxSolrIndexConfig

$searchConfigName=lynxPageSize

$facetSearchConfigName=lynxIndex

$facetSearchConfigDescription=Lynx Solr Index

$searchIndexNamePrefix=lynx

$solrIndexedType=lynxProductType

$indexBaseSite=lynx

$indexLanguages=ja,en,de,zh

$indexCurrencies=USD,CNY

INSERT\_UPDATE SolrFacetSearchConfig;name[unique=true];description;indexNamePrefix;languages(isocode);currencies(isocode);solrServerConfig(name);solrSearchConfig(description);solrIndexConfig(name);solrIndexedTypes(identifier);enabledLanguageFallbackMechanism;$catalogVersions

;$facetSearchConfigName;$facetSearchConfigDescription;$searchIndexNamePrefix;$indexLanguages;$indexCurrencies;$serverConfigName;$searchConfigName;$indexConfigName;$solrIndexedType;true;$productCatalog:Online,$productCatalog:Staged

### 更新BaseSite 查询方式：

UPDATE BaseSite;uid[unique=true];solrFacetSearchConfiguration(name)

;$indexBaseSite;$facetSearchConfigName

### 定义solr字段：

# Non-facet properties

INSERT\_UPDATE SolrIndexedProperty;solrIndexedType(identifier)[unique=true] ;name[unique=true] ;type(code) ;sortableType(code) ;currency[default=false] ;localized[default=false] ;multiValue[default=false] ;useForSpellchecking[default=false] ;useForAutocomplete[default=false] ;fieldValueProvider ;valueProviderParameter

#系统自带属性 ;$solrIndexedType; itemtype ;string ; ; ; ; ; ; ;

;$solrIndexedType; code ;string ; ; ; ; ;true;true;springELValueProvider;code

#自定义属性：

;$solrIndexedType ;lynxVariantColor ;string ; ; ;true ;true ; ; ;lynxProductVariantColorProvider ;

里面有一个provider，是用来对solr中的这个字段进行取值的，有时候会有一些特殊处理，如：字段展示时格式化处理。

### 创建sorl sql.

INSERT\_UPDATE SolrIndexerQuery;solrIndexedType(identifier)[unique=true];identifier[unique=true];type(code);injectCurrentDate[default=true];injectCurrentTime[default=true];injectLastIndexTime[default=true];query;user(uid)

;$solrIndexedType;$searchIndexNamePrefix-fullQuery;full;;;false;"SELECT DISTINCT {PK} FROM

{Product AS p

join LynxColorVariantProduct AS v ON {v.baseProduct}={p.pk}

}

WHERE {p:varianttype} IS NOT NULL";anonymous

;$solrIndexedType;$searchIndexNamePrefix-updateQuery;update;;;;"

SELECT DISTINCT tbl.pk, tbl.code FROM (

{{

SELECT DISTINCT {p:PK} AS pk, {p:code} AS code

FROM {

Product AS p

LEFT JOIN CustomerReview AS cr ON {cr:product}={p:PK}

JOIN LynxColorVariantProduct AS v ON {v.baseProduct}={p.pk}

}

WHERE {p:varianttype} IS NOT NULL AND ({p:modifiedtime} >= ?lastIndexTime OR {cr:modifiedtime} >= ?lastIndexTime)

}}

UNION

{{

SELECT {p:PK} AS pk, {p:code} AS code

FROM {

Product AS p

JOIN LynxColorVariantProduct AS v ON {v.baseProduct}={p.pk}

} WHERE {p:code} IN (

{{

SELECT DISTINCT {sl:productCode} FROM {StockLevel AS sl} WHERE {sl:modifiedtime} >= ?lastIndexTime

}}

)

}}

) tbl ORDER BY tbl.code

";anonymous

### 定义provider:

**public** **class** LynxProductVariantColorProvider **extends** AbstractPropertyFieldValueProvider **implements** FieldValueProvider

{

**private** FieldNameProvider fieldNameProvider;

**private** CommonI18NService commonI18NService;

**private** **static** **final** Logger ***LOG*** = Logger.*getLogger*(LynxProductVariantColorProvider.**class**);

**private** **static** **final** String ***SPLIT*** = "{$}";

@Override

**public** Collection<FieldValue> getFieldValues(**final** IndexConfig indexConfig, **final** IndexedProperty indexedProperty,

**final** Object model) **throws** FieldValueProviderException

{

**final** Collection<FieldValue> fieldValues = **new** ArrayList<FieldValue>();

**final** ProductModel product = (ProductModel) model;

**if** (product == **null**)

{

***LOG***.error("solr get FieldValue product is null");

**return** fieldValues;

}

**final** Collection coll = product.getVariants();

**if** (CollectionUtils.*isEmpty*(coll))

{

***LOG***.error("solr get FileldValue ,product variants is emtpy,can't get variant color properties");

**return** fieldValues;

}

**final** Iterator iter = coll.iterator();

**while** (iter.hasNext())

{

**final** Object obj = iter.next();

**if** (obj **instanceof** LynxColorVariantProductModel)

{

**final** String code = ((LynxColorVariantProductModel) obj).getCode();

**final** LynxProductColorModel colorModel = ((LynxColorVariantProductModel) obj).getColor();

String color = "";

String colorText = "";

**if** (colorModel != **null**)

{

**final** MediaModel media = colorModel.getColorImage();

**if** (media != **null**)

{

color = media.getURL();

}

**if** (indexedProperty.isLocalized())

{

**final** Locale currentLocal = commonI18NService.getLocaleForLanguage(commonI18NService.getCurrentLanguage());

colorText = colorModel.getColor(currentLocal);

}

}

addFieldValues(fieldValues, indexedProperty, code + ***SPLIT*** + color + ***SPLIT*** + colorText);

}

}

**return** fieldValues;

}

**private** **void** addFieldValues(**final** Collection<FieldValue> fieldValues, **final** IndexedProperty indexedProperty,

**final** String colors)

{

**final** Collection<String> fieldNames = getFieldNameProvider().getFieldNames(indexedProperty, **null**);

**for** (**final** String fieldName : fieldNames)

{

fieldValues.add(**new** FieldValue(fieldName, colors));

}

}

Provider用来处理solr里的字段，通过provider的处理，字段才能在存在于solr中。

### Solr字段在查询中如果需要列为可查询字段，必须有如下配置：

<alias name=*"defaultCommerceSearchTextPopulator"* alias=*"commerceSearchTextPopulator"* />

<bean id=*"defaultCommerceSearchTextPopulator"* class=*"de.hybris.platform.commerceservices.search.solrfacetsearch.populators.SearchTextPopulator"*>

<property name=*"freeTextQueryBuilders"*>

<list>

<bean class=*"de.hybris.platform.commerceservices.search.solrfacetsearch.querybuilder.impl.DefaultFreeTextQueryBuilder"*>

<property name=*"propertyName"* value=*"lynxVariantColorText"* />

<property name=*"boost"* value=*"60"* />

</bean>

</list> </property>

</bean>

### Solr在页面展示的数据需要通过如下populator进行处理

**public class LynxSearchResultProductPopulator extends SearchResultProductPopulator{**

**public** **void** populate(**final** SearchResultValueData source, **final** ProductData target) **throws** ConversionException

{

Assert.*notNull*(source, "Parameter source cannot be null.");

Assert.*notNull*(target, "Parameter target cannot be null.");

target.setCode(**this**.<String> getValue(source, "code"));

target.setName(**this**.<String> getValue(source, "name"));

**final** List<String> originPlace = **this**.<List<String>> getValue(source, "lynxOriginPlace\_string\_mv");

**if** (CollectionUtils.*isNotEmpty*(originPlace))

{

target.setOriginPlace(originPlace.iterator().next());

}

}

其中字段lynxOriginPlace\_string\_mv ，他在solr中定义是lynxOriginPlace，但是在solr中查询出来时字段会转化为lynxOriginPlace\_string\_mv ，所以在populator中获取数据时，应该同样使用lynxOriginPlace\_string\_mv 。

## 19Git及Cygwin64 Terminal使用

安装git

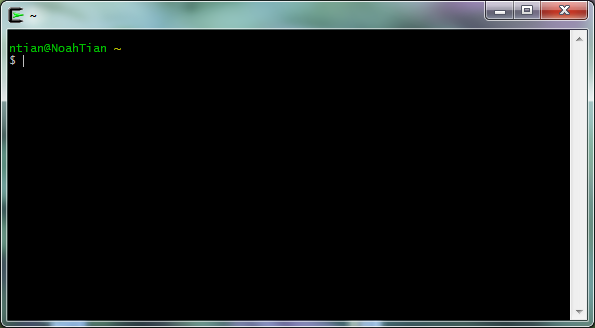
下载地址：

安装cygwin 64:

地址：<https://www.cygwin.com/>

通过cywin来执行git命令。

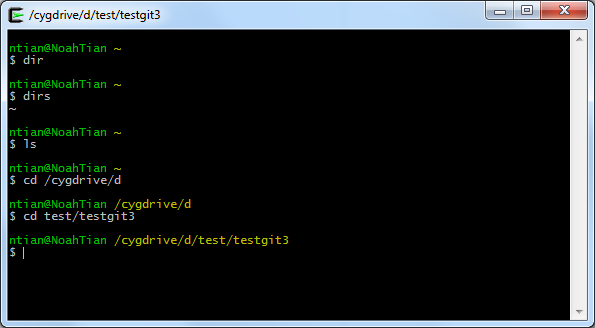
* 打开cywin



默认指向了当前用户的安装目录。

执行命令：**cd /cygdrive/c** 就会进入C盘根目录。执行**cd /cygdrive/d** 进入D盘根目录。

* 新建一个空的文件夹做为git 的目录：

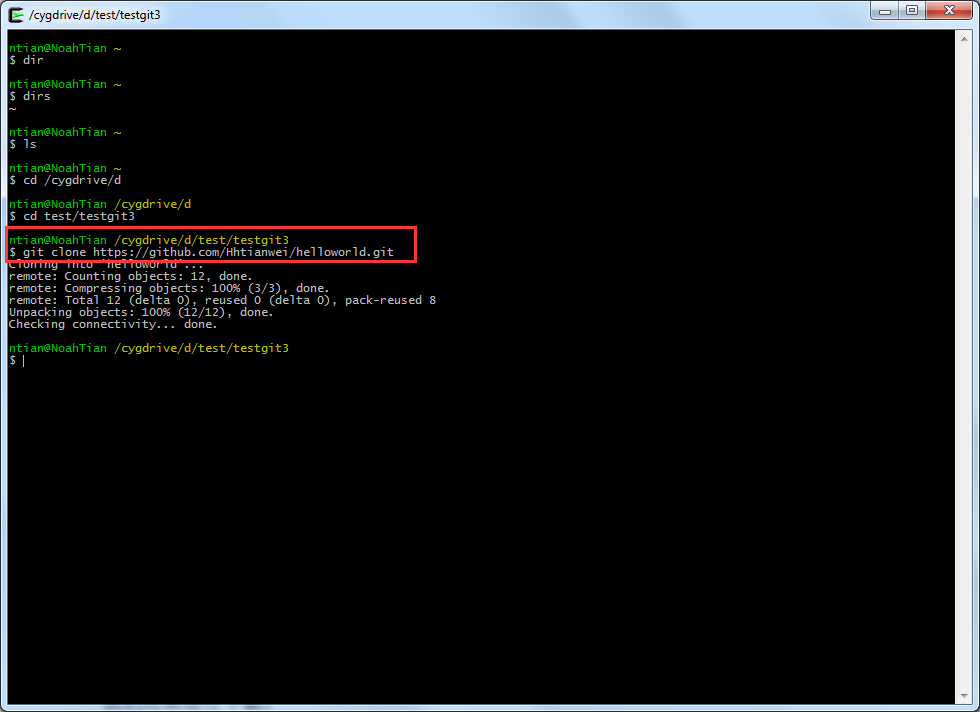


新建了testgit3文件夹，进入到该文件夹。

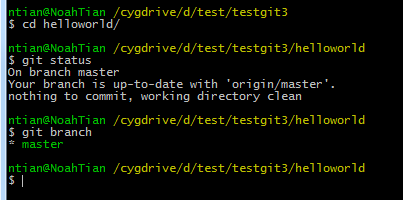
* 因为已经安装了git，所以在这个环境里可以直接使用Git命令。现在开始使用Git命令来进行操作git.对于已经存在的git地址： <https://github.com/Hhtianwei/helloworld.git>

我们对其进行下载。

* 执行命令：**git clone** [**https://github.com/Hhtianwei/helloworld.git**](https://github.com/Hhtianwei/helloworld.git)



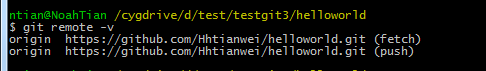
* 执行命令： **cd helloworld**,进入刚刚下载到的git文件夹中：



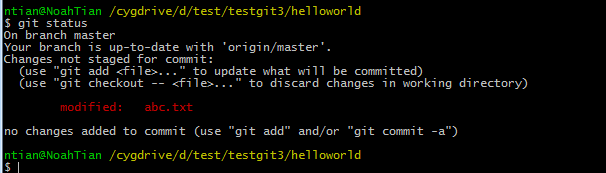
* 执行**git branch** 可以看到所有的Git分支。当前是在master分支上。
* 为git添加远程仓库：

**git remote add origin** [**https://github.com/Hhtianwei/helloworld**](https://github.com/Hhtianwei/helloworld)

查看执行之后的结果 **git remote –v** ：



* 修改git管理 的文件：abc.txt，为其增加新内容.然后使用git 来进行提交。

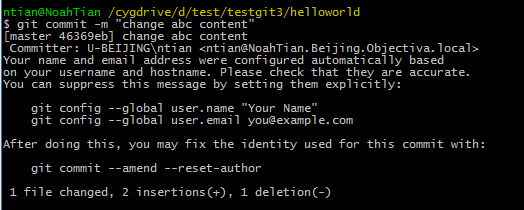


执行**git status**,可以发现abc.txt已经被标记红了。

* 执行**git add abc.txt**将文件加入提交列表。



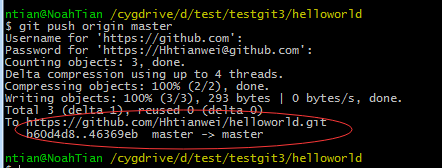
* 执行git commit –m “提交说明”，将文件提交到本地仓库：



* 执行git push origin master将文件提交到远程的master分支上：

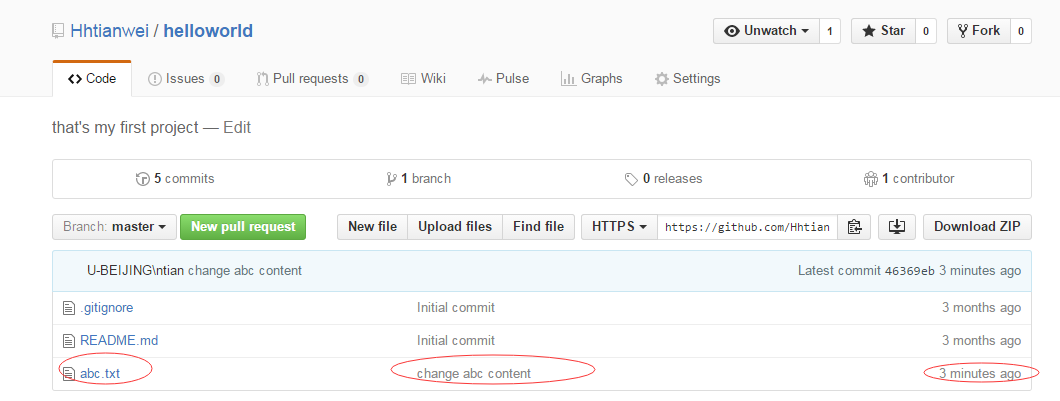


这一步需要输入用户名，密码：

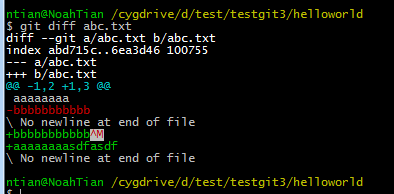


这时候就已经提交完成了。

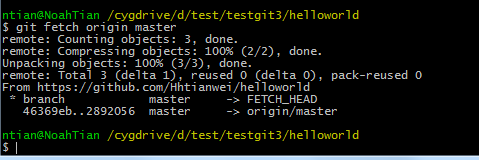
远程gitHub显示已经提交成功了：



* 执行git diff abc.txt可以查看abc文件的修改内容：

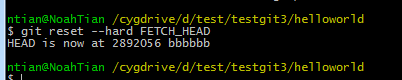


* Git上有文件更新，想要拉代码到本地，可以执行 **git fetch origin master**



* 以上代码就被拉到本地了，但是还没有与当前版本进行合并。如果要应用最新文件，执行：

**Git reset –hard FETCH\_HEAD**



这时，程序 就应用了最新的代码。

## 20. springMVC Ajax方法

Java:

@RequestMapping(value = "/update-password", method = RequestMethod.***POST***)

@ResponseBody

@RequireHardLogIn

**public** String updatePassword(**final** UpdatePasswordForm updatePasswordForm, **final** BindingResult bindingResult, **final** Model model,

**final** RedirectAttributes redirectAttributes) **throws** CMSItemNotFoundException

{

updatePasswordForm这个里面会有数据，前台的form表单里，有这个对象，同时里面字段与这个对象的属性都 一样。

}

Jsp:

|  |
| --- |
| <form:form method=*"post"* commandName=*"updatePasswordForm"* action=*"*${updatePasswordUrl }*"*>  <div class=*"head"*>  <p class=*"myTitle"*>  <spring:theme code=*"account.changepassword.title"* />  </p>  <p class=*"subTitle"*>  <spring:theme code=*"account.changePassword.tips.formtitle"* />  </p>  </div>  <div class=*"inputBox"*> <!-- errorTips -->  <div class=*"commonInputBox"*>  <div class=*"form-group"*>  <label class=*"control-label "* for=*"currentPassword"*>  <spring:theme code=*"account.changePassword.tips.activepw"*/></label>  <spring:theme code=*'account.changePassword.tags.activepw'* var=*"activepw"*/>  <form:password path=*"currentPassword"* cssClass=*"textInput commonInput"* placeholder=*"*${activepw}*"* />  </div>  <span class=*"clear-input"*></span>  <span class=*"errormessage"*>  <form:errors path=*"currentPassword"* cssClass=*"error"*></form:errors>  </span>  </div>  </div>  <!--inputBox-->  <div class=*"inputBox"*>  <div class=*"commonInputBox"*>  <div class=*"form-group"*>  <label class=*"control-label "* for=*"newPassword"*>  <spring:theme code=*"account.changePassword.tips.putnewpw"*/></label>  <spring:theme code=*'account.changePassword.tags.putnewpw'* var=*"putnewpw"*/>  <form:password path=*"newPassword"* cssClass=*"textInput commonInput"* placeholder=*"*${putnewpw}*"* />  </div>  <span class=*"clear-input"*></span>  <span class=*"errormessage"*>  <form:errors path=*"newPassword"* cssClass=*"error"*></form:errors>  </span>  </div>  </div>  <!--inputBox-->  <div class=*"inputBox last"*>  <div class=*"commonInputBox"*>  <div class=*"form-group"*>  <label class=*"control-label "* for=*"checkNewPassword"*>  <spring:theme code=*"account.changePassword.tips.againpw"*/></label>  <spring:theme code=*'account.changePassword.tags.againpw'* var=*"againpw"*/>  <form:password path=*"checkNewPassword"* cssClass=*"textInput commonInput"* placeholder=*"*${againpw}*"* />  </div>  <span class=*"clear-input"*></span>  <span class=*"errormessage"*>  <form:errors path=*"checkNewPassword"* cssClass=*"error"*></form:errors>  </span>  </div>  </div>  <input type=*"button"* class=*"submitBtn"*  value=*"*<spring:theme code=*"account.changepassword.form.submit"*/>*"* />  <div class=*"readMore"*>  <p class=*"title"*><spring:theme code=*"account.changePassword.helpmessage.title"*/></p>  <p class=*"rContent"*>  <spring:theme code=*"account.changePassword.helpmessage.con"* />  </p>  </div>  </form:form> |

**Js:**

|  |
| --- |
| var $form = $('#updatePasswordForm');  $.ajax({  url:url,  type:"post",  data:$form.serialize(),  success:**function**(data){  alert("data:" + data);  console.log(data);  }  }); |

## 21 hybris导入media对象方法.

1. 在指定的文件夹中上传相应的图片。例如：

/icinitialdata/resources/icinitialdata/import/sampledata/icolor/images/banks/jiaotongbank.jpg

将文件jiaotongbank.jpg上传到相应的目录中。

1. 为导入对象设置相应的路径。

|  |
| --- |
| $contentCatalog=icolorContentCatalog |

|  |
| --- |
| $contentCV=catalogVersion(CatalogVersion.catalog(Catalog.id[default=$contentCatalog]),CatalogVersion.version[default=Staged])[default=$contentCatalog:Staged] |

|  |
| --- |
| $imageResource=jar:com.npz.icolor.initialdata.constants.IcInitialDataConstants&/icinitialdata/import/sampledata/icolor/images/banks |

1. 编写相应的导入语句。

|  |
| --- |
| INSERT\_UPDATE Media ;$contentCV[unique=true] ;code[unique=true] ;realfilename ;@media[translator=de.hybris.platform.impex.jalo.media.MediaDataTranslator] ;mime[default='image/jpeg'] ;altText;folder(qualifier)[default='images'] |

1. 加入相应的导入数据，执行导入。

|  |
| --- |
| ; ;jiaotongbank ;jiaotongbank.jpg ;$imageResource/jiaotongbank.jpg ;image/jpg ; ; |

|  |
| --- |
| ; ;zhaoshangbank ;zhongguobank.jpg ;$imageResource/zhongguobank.jpg ;image/jpg ; ; |

## 22 hybris导入对象

1. 导入对象一定要设置unique="true"。设置方式如下：

|  |
| --- |
| <itemtype code="IcolorBankInfo" extends="GenericItem" autocreate="true" generate="true"  jaloclass="com.npz.icolor.core.jalo.IcolorBankInfo">  <deployment table="icolorbankinfo" typecode="27023" />  <attributes>  <attribute autocreate="true" qualifier="bankName" type="java.lang.String" generate="true">  <persistence type="property"/>  <modifiers read="true" write="true" search="true" optional="true" unique="true"/>  </attribute> |

1. 对象中如果有media类型的字段，进行如下操作。

定义media字段的插入类型

|  |
| --- |
| $contentCatalog=icolorContentCatalog  $catalogVersion=catalogversion(catalog(id[default=$contentCatalog]),version[default='Staged'])[unique=true,default=$contentCatalog:Staged]  $image=image(code, $catalogVersion) |

在进行插入操作的时候 ，使用$image代替原来image字段。

1. 对象中有hybris Enum类型的数据，进行如下操作。

定义枚举类型的插入方式

|  |
| --- |
| $type=type(code,itemtype(code)) |

在进行插入操作的时候 ，使用$image代替原来image字段。

插入数据时，枚举类型数据用如下方式进行导入：

|  |
| --- |
| CREDIT\_CARD:IcolorBankTyp |

注 ：CREDIT\_CARD 是在枚举里定义的常量 IcolorBankTyp 是枚举的类名称

1. 导入语句

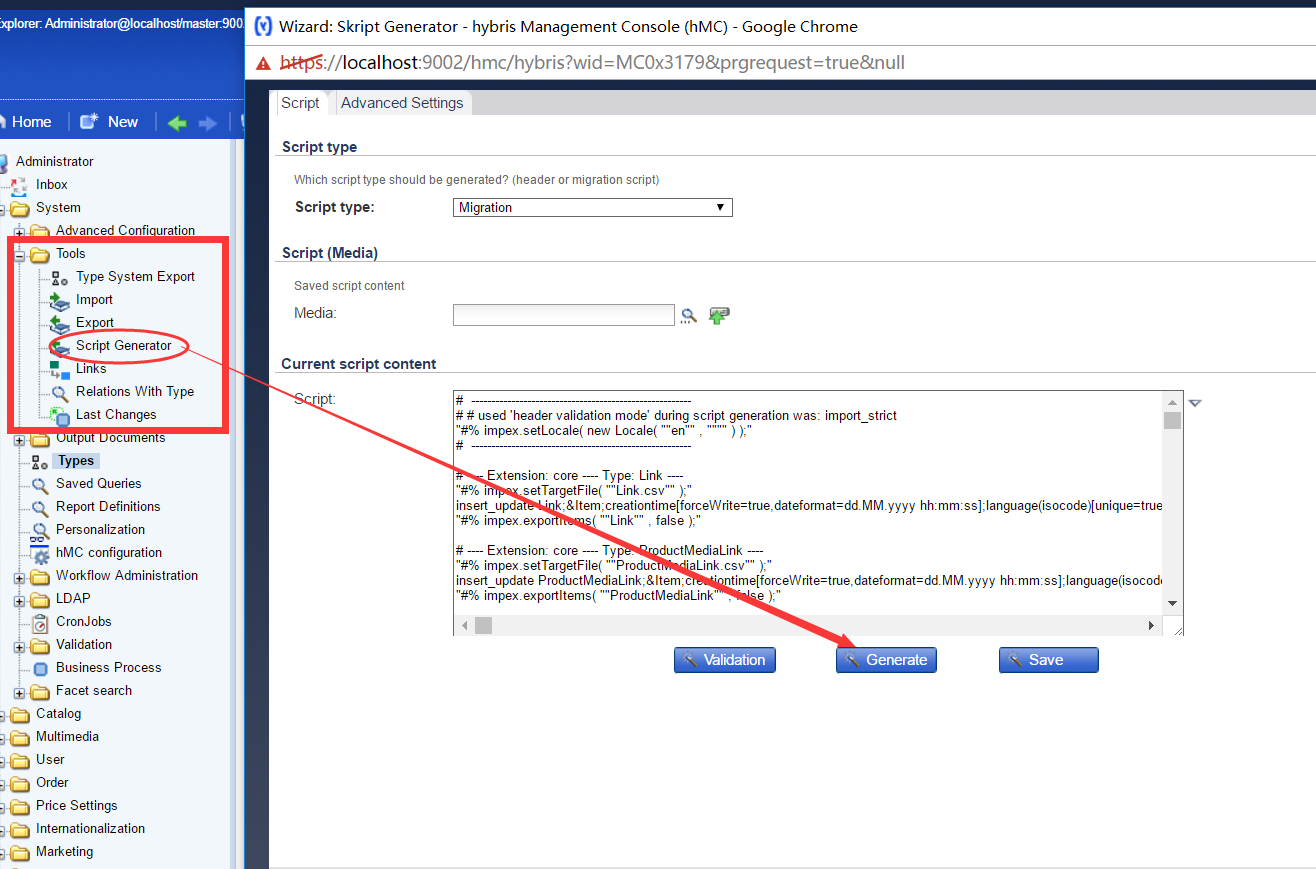
|  |
| --- |
| INSERT\_UPDATE IcolorBankInfo ;bankName[unique=true] ;$image ;limitComments ;isAlive[default = true] ;$type |

1. 导入数据

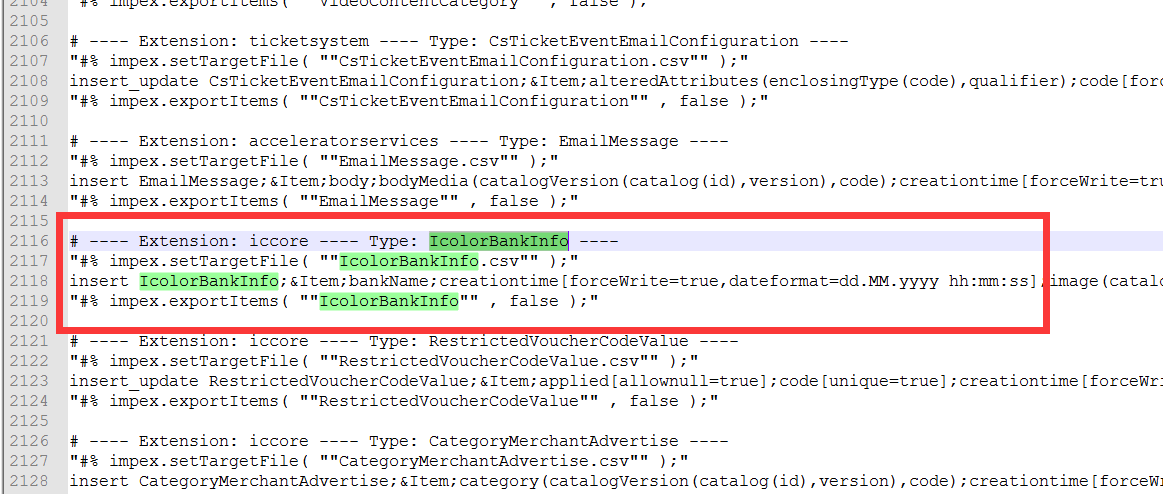
|  |
| --- |
| ;交通银行 ;jiaotongbank ;每日限额5万 ; ;CREDIT\_CARD:IcolorBankType |

如果在进行导入时，不知道如何进行字段格式设置，可以参考以下方式：

1. 打开hmc,通过已经生成的数据语句来确定impex的写法



将生成的数据copy出来，找到相应的对象，例如：



|  |
| --- |
| insert IcolorBankInfo;&Item;bankName;creationtime[forceWrite=true,dateformat=dd.MM.yyyy hh:mm:ss];image(catalogVersion(catalog(id),version),code);isAlive;limitComments;modifiedtime[dateformat=dd.MM.yyyy hh:mm:ss];owner(&Item)[allownull=true];type(code,itemtype(code)) |

该语句就是在进行impex导入数据里，每个字段的格式。如果有list,setmap类型的数据，可以在这里面找到对应。

## 23 jquery中，依据内容对按钮置灰或者启用

1. 当用户在页面里的input框没有输入信息时，按钮默认应该是灰色，且不能被点击，当用户输入有效内容后，按钮才可以点击。

|  |
| --- |
| **var** $addNewCardNextBtn = $("button.addNewCardBtn");  $("form.addNewCardform input[name='name'],form.addNewCardform input[name='accountNo']").on("input",**function**(evt){  **var** name = $("form.addNewCardform input[name='name']").val();  **var** accountNo = $("form.addNewCardform input[name='accountNo']").val();  **var** bank = $("form.addNewCardform input[name='bank']").val();  **if**(name.trim().length && accountNo.trim().length && bank.trim().length){  $addNewCardNextBtn.removeAttr("disabled");  $addNewCardNextBtn.removeClass("disabled");  }**else**{  $addNewCardNextBtn.prop("disabled","disabled");  $addNewCardNextBtn.addClass("disabled");  }  }); |

1. 前提，项目中有disabled这个样式。然后把相应的字段改这真实环境中使用的。
2. 主要使用的是jquery中的input事件进行管理。

## 24 Java http 请求方法

通过java程序来进行http访问，可以访问get,post请求。

HttpClient:

|  |
| --- |
| **package** com.tim.http;  **import** java.io.BufferedReader;  **import** java.io.ByteArrayOutputStream;  **import** java.io.IOException;  **import** java.io.InputStream;  **import** java.io.PrintStream;  **import** java.net.HttpURLConnection;  **import** java.net.MalformedURLException;  **import** java.net.ProtocolException;  **import** java.net.URISyntaxException;  **import** java.net.URL;  **import** java.net.URLConnection;  **import** javax.net.ssl.HttpsURLConnection;  **import** com.tim.http.BaseHttpSSLSocketFactory.TrustAnyHostnameVerifier;  **public** **class** HttpClient {  /\*\*  \* 目标地址  \*/  **private** URL url;  /\*\*  \* 通信连接超时时间  \*/  **private** **int** connectionTimeout;  /\*\*  \* 通信读超时时间  \*/  **private** **int** readTimeOut;  /\*\*  \* 通信结果  \*/  **private** String result;  /\*\*  \* 获取通信结果  \* **@return**  \*/  **public** String getResult() {  **return** result;  }  /\*\*  \* 设置通信结果  \* **@param** result  \*/  **public** **void** setResult(String result) {  **this**.result = result;  }  /\*\*  \* 构造函数  \* **@param** url 目标地址  \* **@param** connectionTimeout HTTP连接超时时间  \* **@param** readTimeOut HTTP读写超时时间  \*/  **public** HttpClient(String url, **int** connectionTimeout, **int** readTimeOut) {  **try** {  **this**.url = **new** URL(url);  **this**.connectionTimeout = connectionTimeout;  **this**.readTimeOut = readTimeOut;  } **catch** (MalformedURLException e) {  }  }  **public** **int** send2(String sendData, String encoding) **throws** Exception {  **try** {  HttpURLConnection httpURLConnection = createConnection(encoding);  **if**(**null** == httpURLConnection){  **throw** **new** Exception("创建联接失败");  }  System.***out***.println("请求报文:[" + sendData + "]");  **this**.requestServer(httpURLConnection, sendData,  encoding);  **this**.result = **this**.response(httpURLConnection, encoding);  System.***out***.println("同步返回报文:[" + result + "]");  **return** httpURLConnection.getResponseCode();  } **catch** (Exception e) {  **throw** e;  }  }  /\*\*  \* 发送信息到服务端 GET方式  \* **@param** data  \* **@param** encoding  \* **@return**  \* **@throws** Exception  \*/  **public** **int** sendGet(String encoding) **throws** Exception {  **try** {  HttpURLConnection httpURLConnection = createConnectionGet(encoding);  **if**(**null** == httpURLConnection){  **throw** **new** Exception("创建联接失败");  }  **this**.result = **this**.response(httpURLConnection, encoding);  System.***out***.println("同步返回报文:[" + result + "]");  **return** httpURLConnection.getResponseCode();  } **catch** (Exception e) {  **throw** e;  }  }  /\*\*  \* HTTP Post发送消息  \*  \* **@param** connection  \* **@param** message  \* **@throws** IOException  \*/  **private** **void** requestServer(**final** URLConnection connection, String message, String encoder)  **throws** Exception {  PrintStream out = **null**;  **try** {  connection.connect();  out = **new** PrintStream(connection.getOutputStream(), **false**, encoder);  out.print(message);  out.flush();  } **catch** (Exception e) {  **throw** e;  } **finally** {  **if** (**null** != out) {  out.close();  }  }  }  /\*\*  \* 显示Response消息  \*  \* **@param** connection  \* **@param** CharsetName  \* **@return**  \* **@throws** URISyntaxException  \* **@throws** IOException  \*/  **private** String response(**final** HttpURLConnection connection, String encoding)  **throws** URISyntaxException, IOException, Exception {  InputStream in = **null**;  StringBuilder sb = **new** StringBuilder(1024);  BufferedReader br = **null**;  **try** {  **if** (200 == connection.getResponseCode()) {  in = connection.getInputStream();  sb.append(**new** String(*read*(in), encoding));  } **else** {  in = connection.getErrorStream();  sb.append(**new** String(*read*(in), encoding));  }  System.***out***.println("HTTP Return Status-Code:["  + connection.getResponseCode() + "]");  **return** sb.toString();  } **catch** (Exception e) {  **throw** e;  } **finally** {  **if** (**null** != br) {  br.close();  }  **if** (**null** != in) {  in.close();  }  **if** (**null** != connection) {  connection.disconnect();  }  }  }    **public** **static** **byte**[] read(InputStream in) **throws** IOException {  **byte**[] buf = **new** **byte**[1024];  **int** length = 0;  ByteArrayOutputStream bout = **new** ByteArrayOutputStream();  **while** ((length = in.read(buf, 0, buf.length)) > 0) {  bout.write(buf, 0, length);  }  bout.flush();  **return** bout.toByteArray();  }    /\*\*  \* 创建连接  \*  \* **@return**  \* **@throws** ProtocolException  \*/  **private** HttpURLConnection createConnection(String encoding) **throws** ProtocolException {  HttpURLConnection httpURLConnection = **null**;  **try** {  httpURLConnection = (HttpURLConnection) url.openConnection();  } **catch** (IOException e) {  System.***out***.println(e.getMessage());  **return** **null**;  }  httpURLConnection.setConnectTimeout(**this**.connectionTimeout);// 连接超时时间  httpURLConnection.setReadTimeout(**this**.readTimeOut);// 读取结果超时时间  httpURLConnection.setDoInput(**true**); // 可读  httpURLConnection.setDoOutput(**true**); // 可写  httpURLConnection.setUseCaches(**false**);// 取消缓存  httpURLConnection.setRequestProperty("Content-type",  "application/x-www-form-urlencoded;charset=" + encoding);  httpURLConnection.setRequestMethod("POST");  **if** ("https".equalsIgnoreCase(url.getProtocol())) {  HttpsURLConnection husn = (HttpsURLConnection) httpURLConnection;  husn.setSSLSocketFactory(**new** BaseHttpSSLSocketFactory());  husn.setHostnameVerifier(**new** TrustAnyHostnameVerifier());//解决由于服务器证书问题导致HTTPS无法访问的情况  **return** husn;  }  **return** httpURLConnection;  }  /\*\*  \* 创建连接  \*  \* **@return**  \* **@throws** ProtocolException  \*/  **private** HttpURLConnection createConnectionGet(String encoding) **throws** ProtocolException {  HttpURLConnection httpURLConnection = **null**;  **try** {  httpURLConnection = (HttpURLConnection) url.openConnection();  } **catch** (IOException e) {  System.***out***.println(e.getMessage());  **return** **null**;  }  httpURLConnection.setConnectTimeout(**this**.connectionTimeout);// 连接超时时间  httpURLConnection.setReadTimeout(**this**.readTimeOut);// 读取结果超时时间  httpURLConnection.setUseCaches(**false**);// 取消缓存  httpURLConnection.setRequestProperty("Content-type",  "application/x-www-form-urlencoded;charset=" + encoding);  httpURLConnection.setRequestMethod("GET");  **if** ("https".equalsIgnoreCase(url.getProtocol())) {  HttpsURLConnection husn = (HttpsURLConnection) httpURLConnection;  husn.setSSLSocketFactory(**new** BaseHttpSSLSocketFactory());  husn.setHostnameVerifier(**new** TrustAnyHostnameVerifier());//解决由于服务器证书问题导致HTTPS无法访问的情况  **return** husn;  }  **return** httpURLConnection;  }  } |

调用请求：

|  |
| --- |
| **package** com.tim.http;  **import** java.util.HashMap;  **import** java.util.Map;  **public** **class** HttpService {  **public** **static** Map<String, String> post2(String reqData, String reqUrl, String encoding) {  Map<String, String> rspData = **new** HashMap<String, String>();  // 发送后台请求数据  HttpClient hc = **new** HttpClient(reqUrl, 30000, 30000);  **try** {  **int** status = hc.send2(reqData, encoding);  **if** (200 == status) {  String resultString = hc.getResult();  **if** (**null** != resultString && !"".equals(resultString)) {  System.***out***.println("resultString:" + resultString);  } **else** {  System.***out***.println("=============error==============");  }  } **else** {  }  } **catch** (Exception e) {  }  **return** rspData;  }  **public** **static** String get2(String reqUrl, String encoding) {  // 发送后台请求数据  HttpClient hc = **new** HttpClient(reqUrl, 30000, 30000);  **try** {  **int** status = hc.sendGet(encoding);  **if** (200 == status) {  String resultString = hc.getResult();  **if** (**null** != resultString && !"".equals(resultString)) {  **return** resultString;  }  } **else** {  }  } **catch** (Exception e) {  }  **return** **null**;  }  **public** **static** **void** main(String[] args) {  String queryParams = "order\_no=00050001&mer\_id=310310189120393&start\_date=20161222&end\_date=20161223&dsorder\_no=&&settle\_flag=&settle\_date=&mer\_name=&mac=51E977B8587E69FB04EFDD13B8571650";  String url3 = "http://210.22.91.77:24111/queryService/billqueryServlet";  *post2*(queryParams, url3, "utf-8");  }  } |

处理Https请求

|  |
| --- |
| package com.tim.http;  import java.io.IOException;  import java.net.InetAddress;  import java.net.Socket;  import java.net.UnknownHostException;  import java.security.cert.X509Certificate;  import javax.net.ssl.HostnameVerifier;  import javax.net.ssl.SSLContext;  import javax.net.ssl.SSLSession;  import javax.net.ssl.SSLSocketFactory;  import javax.net.ssl.TrustManager;  import javax.net.ssl.X509TrustManager;  public class BaseHttpSSLSocketFactory extends SSLSocketFactory {  private SSLContext getSSLContext() {  return createEasySSLContext();  }  @Override  public Socket createSocket(InetAddress arg0, int arg1, InetAddress arg2,  int arg3) throws IOException {  return getSSLContext().getSocketFactory().createSocket(arg0, arg1,  arg2, arg3);  }  @Override  public Socket createSocket(String arg0, int arg1, InetAddress arg2, int arg3)  throws IOException, UnknownHostException {  return getSSLContext().getSocketFactory().createSocket(arg0, arg1,  arg2, arg3);  }  @Override  public Socket createSocket(InetAddress arg0, int arg1) throws IOException {  return getSSLContext().getSocketFactory().createSocket(arg0, arg1);  }  @Override  public Socket createSocket(String arg0, int arg1) throws IOException,  UnknownHostException {  return getSSLContext().getSocketFactory().createSocket(arg0, arg1);  }  @Override  public String[] getSupportedCipherSuites() {  // TODO Auto-generated method stub  return null;  }  @Override  public String[] getDefaultCipherSuites() {  // TODO Auto-generated method stub  return null;  }  @Override  public Socket createSocket(Socket arg0, String arg1, int arg2, boolean arg3)  throws IOException {  return getSSLContext().getSocketFactory().createSocket(arg0, arg1,  arg2, arg3);  }  private SSLContext createEasySSLContext() {  try {  SSLContext context = SSLContext.getInstance("SSL");  context.init(null,  new TrustManager[] { MyX509TrustManager.manger }, null);  return context;  } catch (Exception e) {  System.out.println(e);  return null;  }  }  public static class MyX509TrustManager implements X509TrustManager {  static MyX509TrustManager manger = new MyX509TrustManager();  public MyX509TrustManager() {  }  public X509Certificate[] getAcceptedIssuers() {  return null;  }  public void checkClientTrusted(X509Certificate[] chain, String authType) {  }  public void checkServerTrusted(X509Certificate[] chain, String authType) {  }  }  public static class TrustAnyHostnameVerifier implements HostnameVerifier {  public boolean verify(String hostname, SSLSession session) {  return true;  }  }  } |

## 25 Java解析xml

1. 使用dom4j

|  |
| --- |
| **import** org.dom4j.Document;  **import** org.dom4j.DocumentException;  **import** org.dom4j.DocumentHelper;  **import** org.dom4j.Element;  Document doc = **null**;  doc = DocumentHelper.*parseText*(context);  **final** Element requestDOC = doc.getRootElement();  **final** List<Element> root2 = requestDOC.elements();  **for** (**final** Iterator it = root2.iterator(); it.hasNext();)  {  **final** Element elm = (Element) it.next();  **final** List<Element> es = elm.elements();  **for** (**final** Element e : es)  {  **if** (POSConstants.***CALLBACK\_HEADER\_VERSION***.equals(e.getName()))  {    responseVersion.setText(e.getText());  **continue**;  }  }  } |

Xml对象转java字串，直接使用：

Document respDoc

respDoc.asXML();

生成的string包含xml头信息。

如果使用doc.getRootElement().asXML();生成的xml字串不包含头信息。

## 26 Java解析json

1. Json字串转json对象

|  |
| --- |
| **import** atg.taglib.json.util.JSONArray;  **import** atg.taglib.json.util.JSONException;  **import** atg.taglib.json.util.JSONObject;  JSONObject json = **null**;  **try**  {  json = **new** JSONObject(result);  }  **catch** (**final** JSONException e)  {  ***LOG***.error(String.*format*("json string %s convert to json object error", result), e);  }  **final** JSONArray array = json.getJSONArray(POSConstants.***QUERY\_RESULT\_LIST***);  **if** (array == **null** || array.length() < 1)  {  **return** **null**;  }  **final** JSONObject object = array.getJSONObject(0);  **final** IcPaymentInforModel model = getModelService().create(IcPaymentInforModel.**class**);  **if** (object.has(POSConstants.***QUERY\_RESULT\_TRACE\_TIME***))  {  object.getString(POSConstants.***QUERY\_RESULT\_TRACE\_TIME***)  } |

## 27 Java生成二维码

1. 调用servlet在jsp页面生成二维码图片

|  |
| --- |
| **package** test0200;  **import** java.awt.image.BufferedImage;  **import** java.io.File;  **import** java.io.IOException;  **import** javax.imageio.ImageIO;  **import** javax.servlet.ServletException;  **import** javax.servlet.http.HttpServlet;  **import** javax.servlet.http.HttpServletRequest;  **import** javax.servlet.http.HttpServletResponse;  **public** **class** TestServlet **extends** HttpServlet{  @Override  **protected** **void** doPost(HttpServletRequest req, HttpServletResponse response) **throws** ServletException, IOException {  StringBuilder sb = **new** StringBuilder();  sb.append("noah,");  sb.append("objectiva,");  sb.append(System.*currentTimeMillis*());  //sb.append("http://www.baidu.com");  System.***out***.println("==============="+sb.toString()+"=================");  String path = req.getServletContext().getRealPath("/img"+File.***separator***+"test01.png");  BufferedImage image = **null**;  **try** {  image = QRCodeUtil.*createImage*(sb.toString(), path, **true**);  } **catch** (Exception e) {  }  image.flush();  ImageIO.*write*(image, "jpg", response.getOutputStream());  }    @Override  **protected** **void** doGet(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {  **this**.doPost(req, resp);  }  } |

1. 接收页面代码

|  |
| --- |
| <img alt=*""* id=*"img"* src=*"*<%=basePath %>*/twoDimensional"*/>  <button id="btn">reloadImage</button> |

1. 更新二维码

|  |
| --- |
| $(**function**(){  $img = $("#img");  $btn = $("#btn");  $btn.on("click",**function**(){  $img.attr("src","/test0200/twoDimensional?r="+Math.random());  });  }); |

1. 生成二维码方法

|  |
| --- |
| /\*\*  \*  \*/  **package** test0200;  **import** java.awt.BasicStroke;  **import** java.awt.Graphics;  **import** java.awt.Graphics2D;  **import** java.awt.Image;  **import** java.awt.Shape;  **import** java.awt.geom.RoundRectangle2D;  **import** java.awt.image.BufferedImage;  **import** java.io.File;  **import** java.io.OutputStream;  **import** java.util.Hashtable;  **import** java.util.Random;  **import** javax.imageio.ImageIO;  **import** com.google.zxing.BarcodeFormat;  **import** com.google.zxing.BinaryBitmap;  **import** com.google.zxing.DecodeHintType;  **import** com.google.zxing.EncodeHintType;  **import** com.google.zxing.MultiFormatReader;  **import** com.google.zxing.MultiFormatWriter;  **import** com.google.zxing.Result;  **import** com.google.zxing.common.BitMatrix;  **import** com.google.zxing.common.HybridBinarizer;  **import** com.google.zxing.qrcode.decoder.ErrorCorrectionLevel;  /\*\*  \* **@author** Administrator  \*  \*/  **public** **class** QRCodeUtil  {  **private** **static** **final** String ***CHARSET*** = "utf-8";  **private** **static** **final** String ***FORMAT\_NAME*** = "JPG";  // 二维码尺寸  **private** **static** **final** **int** ***QRCODE\_SIZE*** = 300;  // LOGO宽度  **private** **static** **final** **int** ***WIDTH*** = 60;  // LOGO高度  **private** **static** **final** **int** ***HEIGHT*** = 60;  **public** **static** BufferedImage createImage(**final** String content, **final** String imgPath, **final** **boolean** needCompress)  **throws** Exception  {  **final** Hashtable<EncodeHintType, Object> hints = **new** Hashtable<EncodeHintType, Object>();  hints.put(EncodeHintType.***ERROR\_CORRECTION***, ErrorCorrectionLevel.***H***);  hints.put(EncodeHintType.***CHARACTER\_SET***, ***CHARSET***);  hints.put(EncodeHintType.***MARGIN***, 1);  **final** BitMatrix bitMatrix = **new** MultiFormatWriter().encode(content, BarcodeFormat.***QR\_CODE***, ***QRCODE\_SIZE***, ***QRCODE\_SIZE***, hints);  **final** **int** width = bitMatrix.getWidth();  **final** **int** height = bitMatrix.getHeight();  **final** BufferedImage image = **new** BufferedImage(width, height, BufferedImage.***TYPE\_INT\_RGB***);  **for** (**int** x = 0; x < width; x++)  {  **for** (**int** y = 0; y < height; y++)  {  image.setRGB(x, y, bitMatrix.get(x, y) ? 0xFF000000 : 0xFFFFFFFF);  }  }  **if** (imgPath == **null** || "".equals(imgPath))  {  **return** image;  }  // 插入图片  QRCodeUtil.*insertImage*(image, imgPath, needCompress);  **return** image;  }  /\*\*  \* 插入LOGO  \*  \* **@param** source  \* 二维码图片  \* **@param** imgPath  \* LOGO图片地址  \* **@param** needCompress  \* 是否压缩  \* **@throws** Exception  \*/  **private** **static** **void** insertImage(**final** BufferedImage source, **final** String imgPath, **final** **boolean** needCompress) **throws** Exception  {  **final** File file = **new** File(imgPath);  **if** (!file.exists())  {  System.***err***.println("" + imgPath + " 该文件不存在！");  **return**;  }  Image src = ImageIO.*read*(**new** File(imgPath));  **int** width = src.getWidth(**null**);  **int** height = src.getHeight(**null**);  **if** (needCompress)  { // 压缩LOGO  **if** (width > ***WIDTH***)  {  width = ***WIDTH***;  }  **if** (height > ***HEIGHT***)  {  height = ***HEIGHT***;  }  **final** Image image = src.getScaledInstance(width, height, Image.***SCALE\_SMOOTH***);  **final** BufferedImage tag = **new** BufferedImage(width, height, BufferedImage.***TYPE\_INT\_RGB***);  **final** Graphics g = tag.getGraphics();  g.drawImage(image, 0, 0, **null**); // 绘制缩小后的图  g.dispose();  src = image;  }  // 插入LOGO  **final** Graphics2D graph = source.createGraphics();  **final** **int** x = (***QRCODE\_SIZE*** - width) / 2;  **final** **int** y = (***QRCODE\_SIZE*** - height) / 2;  graph.drawImage(src, x, y, width, height, **null**);  **final** Shape shape = **new** RoundRectangle2D.Float(x, y, width, width, 6, 6);  graph.setStroke(**new** BasicStroke(3f));  graph.draw(shape);  graph.dispose();  }  /\*\*  \* 生成二维码(内嵌LOGO)  \*  \* **@param** content  \* 内容  \* **@param** imgPath  \* LOGO地址  \* **@param** destPath  \* 存放目录  \* **@param** needCompress  \* 是否压缩LOGO  \* **@throws** Exception  \*/  **public** **static** **void** encode(**final** String content, **final** String imgPath, **final** String destPath, **final** **boolean** needCompress)  **throws** Exception  {  **final** BufferedImage image = QRCodeUtil.*createImage*(content, imgPath, needCompress);  *mkdirs*(destPath);  **final** String file = **new** Random().nextInt(99999999) + ".jpg";  ImageIO.*write*(image, ***FORMAT\_NAME***, **new** File(destPath + "/" + file));  }  **public** **static** String encode\_fiximgname(**final** String content, **final** String imgPath, **final** String destPath,  **final** **boolean** needCompress) **throws** Exception  {  **final** BufferedImage image = QRCodeUtil.*createImage*(content, imgPath, needCompress);  **final** String path = destPath.substring(0, destPath.lastIndexOf("\\"));  **final** String file = destPath.substring(destPath.lastIndexOf("\\") + 1);  *mkdirs*(path);  ImageIO.*write*(image, ***FORMAT\_NAME***, **new** File(path + "\\" + file));  **return** file;  }  /\*\*  \* 当文件夹不存在时，mkdirs会自动创建多层目录，区别于mkdir．(mkdir如果父目录不存在则会抛出异常)  \*  \* **@author** lanyuan Email: mmm333zzz520@163.com  \* **@date** 2013-12-11 上午10:16:36  \* **@param** destPath  \* 存放目录  \*/  **public** **static** **void** mkdirs(**final** String destPath)  {  **final** File file = **new** File(destPath);  //当文件夹不存在时，mkdirs会自动创建多层目录，区别于mkdir．(mkdir如果父目录不存在则会抛出异常)  **if** (!file.exists() && !file.isDirectory())  {  file.mkdirs();  }  }  /\*\*  \* 生成二维码(内嵌LOGO)  \*  \* **@param** content  \* 内容  \* **@param** imgPath  \* LOGO地址  \* **@param** destPath  \* 存储地址  \* **@throws** Exception  \*/  **public** **static** **void** encode(**final** String content, **final** String imgPath, **final** String destPath) **throws** Exception  {  QRCodeUtil.*encode*(content, imgPath, destPath, **false**);  }  /\*\*  \* 生成二维码  \*  \* **@param** content  \* 内容  \* **@param** destPath  \* 存储地址  \* **@param** needCompress  \* 是否压缩LOGO  \* **@throws** Exception  \*/  **public** **static** **void** encode(**final** String content, **final** String destPath, **final** **boolean** needCompress) **throws** Exception  {  QRCodeUtil.*encode*(content, **null**, destPath, needCompress);  }  /\*\*  \* 生成二维码  \*  \* **@param** content  \* 内容  \* **@param** destPath  \* 存储地址  \* **@throws** Exception  \*/  **public** **static** **void** encode(**final** String content, **final** String destPath) **throws** Exception  {  QRCodeUtil.*encode*(content, **null**, destPath, **false**);  }  **public** **static** String encode\_fiximgname(**final** String content, **final** String destPath) **throws** Exception  {  **return** QRCodeUtil.*encode\_fiximgname*(content, **null**, destPath, **false**);  }  /\*\*  \* 生成二维码(内嵌LOGO)  \*  \* **@param** content  \* 内容  \* **@param** imgPath  \* LOGO地址  \* **@param** output  \* 输出流  \* **@param** needCompress  \* 是否压缩LOGO  \* **@throws** Exception  \*/  **public** **static** **void** encode(**final** String content, **final** String imgPath, **final** OutputStream output, **final** **boolean** needCompress)  **throws** Exception  {  **final** BufferedImage image = QRCodeUtil.*createImage*(content, imgPath, needCompress);  ImageIO.*write*(image, ***FORMAT\_NAME***, output);  }  /\*\*  \* 生成二维码  \*  \* **@param** content  \* 内容  \* **@param** output  \* 输出流  \* **@throws** Exception  \*/  **public** **static** **void** encode(**final** String content, **final** OutputStream output) **throws** Exception  {  QRCodeUtil.*encode*(content, **null**, output, **false**);  }  /\*\*  \* 解析二维码  \*  \* **@param** file  \* 二维码图片  \* **@throws** Exception  \*/  **public** **static** String decode(**final** File file) **throws** Exception  {  BufferedImage image;  image = ImageIO.*read*(file);  **if** (image == **null**)  {  **return** **null**;  }  **final** BufferedImageLuminanceSource source = **new** BufferedImageLuminanceSource(image);  **final** BinaryBitmap bitmap = **new** BinaryBitmap(**new** HybridBinarizer(source));  Result result;  **final** Hashtable<DecodeHintType, Object> hints = **new** Hashtable<DecodeHintType, Object>();  hints.put(DecodeHintType.***CHARACTER\_SET***, ***CHARSET***);  result = **new** MultiFormatReader().decode(bitmap, hints);  **final** String resultStr = result.getText();  **return** resultStr;  }  /\*\*  \* 解析二维码  \*  \* **@param** path  \* 二维码图片地址  \* **@throws** Exception  \*/  **public** **static** String decode(**final** String path) **throws** Exception  {  **return** QRCodeUtil.*decode*(**new** File(path));  }  **public** **static** **void** main(**final** String[] args) **throws** Exception  {  **final** String text = "http://www.yihaomen.com";  QRCodeUtil.*encode*(text, "c:/me.jpg", "c:/barcode", **true**);  }  } |

1. 二维码工具类

|  |
| --- |
| package test0200;  import java.awt.Graphics2D;  import java.awt.geom.AffineTransform;  import java.awt.image.BufferedImage;  import com.google.zxing.LuminanceSource;  public class BufferedImageLuminanceSource extends LuminanceSource {  private final BufferedImage image;  private final int left;  private final int top;  public BufferedImageLuminanceSource(BufferedImage image) {  this(image, 0, 0, image.getWidth(), image.getHeight());  }  public BufferedImageLuminanceSource(BufferedImage image, int left,  int top, int width, int height) {  super(width, height);  int sourceWidth = image.getWidth();  int sourceHeight = image.getHeight();  if (left + width > sourceWidth || top + height > sourceHeight) {  throw new IllegalArgumentException(  "Crop rectangle does not fit within image data.");  }  for (int y = top; y < top + height; y++) {  for (int x = left; x < left + width; x++) {  if ((image.getRGB(x, y) & 0xFF000000) == 0) {  image.setRGB(x, y, 0xFFFFFFFF); // = white  }  }  }  this.image = new BufferedImage(sourceWidth, sourceHeight,  BufferedImage.TYPE\_BYTE\_GRAY);  this.image.getGraphics().drawImage(image, 0, 0, null);  this.left = left;  this.top = top;  }    public byte[] getRow(int y, byte[] row) {  if (y < 0 || y >= getHeight()) {  throw new IllegalArgumentException(  "Requested row is outside the image: " + y);  }  int width = getWidth();  if (row == null || row.length < width) {  row = new byte[width];  }  image.getRaster().getDataElements(left, top + y, width, 1, row);  return row;  }    public byte[] getMatrix() {  int width = getWidth();  int height = getHeight();  int area = width \* height;  byte[] matrix = new byte[area];  image.getRaster().getDataElements(left, top, width, height, matrix);  return matrix;  }    public boolean isCropSupported() {  return true;  }    public LuminanceSource crop(int left, int top, int width, int height) {  return new BufferedImageLuminanceSource(image, this.left + left,  this.top + top, width, height);  }    public boolean isRotateSupported() {  return true;  }    public LuminanceSource rotateCounterClockwise() {  int sourceWidth = image.getWidth();  int sourceHeight = image.getHeight();  AffineTransform transform = new AffineTransform(0.0, -1.0, 1.0,  0.0, 0.0, sourceWidth);  BufferedImage rotatedImage = new BufferedImage(sourceHeight,  sourceWidth, BufferedImage.TYPE\_BYTE\_GRAY);  Graphics2D g = rotatedImage.createGraphics();  g.drawImage(image, transform, null);  g.dispose();  int width = getWidth();  return new BufferedImageLuminanceSource(rotatedImage, top,  sourceWidth - (left + width), getHeight(), width);  }  } |

## 28 每隔一定时间可以重新点击的样式

获取短信验证码时，第一次点击之后，会产生一个倒计时。只有到计时归零之后，以可以进行第二次，第三次的获取。

1. jsp页面

|  |
| --- |
| <%@ page language=*"java"* contentType=*"text/html; charset=utf-8"*  pageEncoding=*"utf-8"*%>    <%  String path = request.getContextPath();  String basePath = request.getScheme() + "://" + request.getServerName() + ":" + request.getServerPort() + path + "/";  %>  <!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=utf-8"*>  <title>Insert title here</title>  <script type=*"text/javascript"* src=*"jquery-1.10.0.js"*></script>  <script type=*"text/javascript"*>  **function** setTime() {  **var** countdown = $(".countdown").text();  console.info("====="+countdown+"========");  **if** (countdown == 0) {  $(".countdown").hide();  $(".defaultSMSTitle").hide();  $(".reSendTitle").show();  $("a.recount").addClass("timeout\_resend");  } **else** {  countdown--;  $(".countdown").text(countdown);  setTimeout("setTime()",1000);  }  }    $(**function**(){  setTime();  $("a.recount").on("click",**function**(){    **var** $span = $(".countdown");    **var** count = $span.attr("attr-data");    $span.text(count);    $(".countdown").show();  $(".defaultSMSTitle").show();  $(".reSendTitle").hide();  $("a.recount").removeClass("timeout\_resend");  setTime();  });  });    </script>  </head>  <body>  <div class=*"col-xs-4 gr-notpadding"*>  <input type=*"text"* name=*"code"*/><a href=*"javascript:;"* class=*"btn btn-block pas-verification recount"*><span attr-data=*"5"* class=*"countdown"*>5</span>  <span class=*"defaultSMSTitle"*>秒后重新发送</span>  <span class=*"reSendTitle"* style="display:*none*">重新发送</span>  </a>  </div>  </body>  </html> |

1. 引用 的js : jquery-1.10.0.js

## 29 java MD5 加密数据

MD5工具类

|  |
| --- |
| **package** test.md5;  **import** java.security.MessageDigest;  **public** **class** MD5Util {  **public** **final** **static** String MD5(String s) {  **char** hexDigits[]={'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};  **try** {  **byte**[] btInput = s.getBytes();  // 获得MD5摘要算法的 MessageDigest 对象  MessageDigest mdInst = MessageDigest.*getInstance*("MD5");  // 使用指定的字节更新摘要  mdInst.update(btInput);  // 获得密文  **byte**[] md = mdInst.digest();  // 把密文转换成十六进制的字符串形式  **int** j = md.length;  **char** str[] = **new** **char**[j \* 2];  **int** k = 0;  **for** (**int** i = 0; i < j; i++) {  **byte** byte0 = md[i];  str[k++] = hexDigits[byte0 >>> 4 & 0xf];  str[k++] = hexDigits[byte0 & 0xf];  }  **return** **new** String(str);  } **catch** (Exception e) {  e.printStackTrace();  **return** **null**;  }  }    **public** **final** **static** String MD5(**byte**[] btInput) {  **char** hexDigits[]={'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};  **try** {  // 获得MD5摘要算法的 MessageDigest 对象  MessageDigest mdInst = MessageDigest.*getInstance*("MD5");  // 使用指定的字节更新摘要  mdInst.update(btInput);  // 获得密文  **byte**[] md = mdInst.digest();  // 把密文转换成十六进制的字符串形式  **int** j = md.length;  **char** str[] = **new** **char**[j \* 2];  **int** k = 0;  **for** (**int** i = 0; i < j; i++) {  **byte** byte0 = md[i];  str[k++] = hexDigits[byte0 >>> 4 & 0xf];  str[k++] = hexDigits[byte0 & 0xf];  }  **return** **new** String(str);  } **catch** (Exception e) {  e.printStackTrace();  **return** **null**;  }  }    **public** **static** **void** main(String[] args) {  System.***out***.println(MD5Util.*MD5*("admin"));  System.***out***.println(MD5Util.*MD5*("加密"));  }  } |

Demo 生成file的MD5值:

|  |
| --- |
| **private** **static** String getMD5File(File p1) **throws** IOException {  InputStream input = **new** FileInputStream(p1);  **byte**[] bs = **new** **byte**[(**int**)p1.length()];  input.read(bs);  **return** MD5Util.*MD5*(bs);  } |

## 30 java复制文件及文件夹

1.copy file方法（文件越大，效率越高）

|  |
| --- |
| **private** **static** **void** copyFile(File source,File target) **throws** IOException{  System.***out***.println("=============start copy file :" + source.getName()+"=========");  FileInputStream input = **new** FileInputStream(source);  FileOutputStream output = **new** FileOutputStream(target);  FileChannel in = input.getChannel();  FileChannel out = output.getChannel();  in.transferTo(0, in.size(), out);    **if**(input != **null**){  input.close();  }  **if**(in != **null**){  in.close();  }  **if**(output != **null**){  output.close();  }  **if**(out != **null**){  out.close();  }  System.***out***.println("===========end copy file :" + source.getName()+"======================");  } |

2.读取文件夹及里面文件

|  |
| --- |
| **private** **static** **void** copyFiles(String source,String target) **throws** IOException{  File sourceFile = **new** File(source);  String size = *getFileSize*(sourceFile.length());  System.***out***.println(String.*format*("file %s size is %s", source,size));  **if**(!sourceFile.exists()){  System.***out***.println(String.*format*("source file %s is not exits",source));  **return** ;  }  File targetFile = **new** File(target);  **if**(!targetFile.exists()){  targetFile.mkdirs();  }    File[] files = sourceFile.listFiles();  **if**(files == **null** || files.length == 0){  System.***out***.print("file directory is not include any file");  **return** ;  }    **for**(File file:files){  **if**(file.isDirectory()){  *copyFiles*(file.getPath(),target + File.***separator*** + file.getName());  }  **if**(file.isFile()){  *copyFile*(file,**new** File(target + File.***separator*** + file.getName()));  }  }  } |

1. 工具方法 ，显示文件大小

|  |
| --- |
| **private** **static** String getFileSize(**long** length) {  **if**(length <1024){  **return** length + "KB";  }  **double** d1 = (**double**)length / 1024;  **if**(d1 < 1024){  **return** **new** java.text.DecimalFormat("#.00").format(d1) + "MB";  }  **double** d2 = d1 / 1024;  **return** **new** java.text.DecimalFormat("#.00").format(d2) + "GB";  } |

## 31. hybris中FlexibleSearch 特殊用法

查询语句中使用between …and

**Eg:**

**SELECT {O.PK} FROM {ORDER AS O} WHERE {O.PK} BETWEEN 8796158591021 AND 87961586223789**

**在程序里，可以通过 ? 传参的方式完成：**

**SELECT {O.PK} FROM {ORDER AS O LEFT JOIN icPaymentInfor AS IP ON {IP.PK}={O.icPaymentInfor} } WHERE {O.DATE} between ?dateStart and ?dateEnd AND {IP.PAY\_TYPE} in('POS','ali2')**

**def sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");**

**def dateStart = sdf.parse("2016-12-21 12:13:14");**

**def dateEnd = sdf.parse("2017-01-09 12:13:14");**

**params.put("dateStart", dateStart);**

**params.put("dateEnd", dateEnd);**

## 32. hac中查询某表时间范围的内容

目前hybris - hac中进行查询订单时，无法指定日期进行查询（与数据库使用oracle有关），为解决这个问题，可以使用字符截取的方式完成日期的指定

Eg:

SELECT {O.PK} FROM {ORDER AS O LEFT JOIN icPaymentInfor AS IP ON {IP.PK}={O.icPaymentInfor} } WHERE substr({O.DATE},0,2)='11' AND substr({O.DATE},4,2)='1月' AND substr({O.DATE},8,2)='17' AND {IP.PAY\_TYPE}='POS'

也可以在script中通过脚本进行执行Eg:

|  |
| --- |
| import java.text.SimpleDateFormat;  //目前的支付类型有三种[ali2,POS,wx3]，如果需要查询某一种类型的订单，需要将以下语句结尾处 POS 修改为相应的订单类型  def FIND\_ORDERS\_QUERY = "SELECT {O.PK} FROM {ORDER AS O LEFT JOIN icPaymentInfor AS IP ON {IP.PK}={O.icPaymentInfor} } WHERE substr({O.DATE},0,2)=?day AND substr({O.DATE},4,2)=?month AND substr({O.DATE},8,2)=?year AND {IP.PAY\_TYPE}='POS'";  def params = new HashMap<String, String>();  //输入日期 不足两位 加 0  params.put("day", "08");    //替换月份 直接写 3月，12月  params.put("month", "1月");    //替换年份 2016年就用 16  params.put("year", "17");  def searchResult = flexibleSearchService.search(FIND\_ORDERS\_QUERY, params);  def list = searchResult.getResult();  if(list != null && list.size() > 0){  //println(list.code);    def sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");  def index = 0;  for(order in list){  println("NO."+ (++index) + " " + order.code + " " + sdf.format(order.date) );  }    }else{  println("there is no orders");  } |

## 33. CMS-COCKPIT在hybris中的使用

1. Impex中加入template

|  |
| --- |
| #PageTemplate  INSERT\_UPDATE PageTemplate ;$contentCV[unique=true] ;uid[unique=true] ;name ;frontendTemplateName ;restrictedPageTypes(code) ;active[default=true] ;velocityTemplate  ; ;test01PageTemplate ;test 01 PageTemplate ;icolor/jiapin/test01Page ;CategoryPage,ContentPage ; ; |

1. 加入ContentPage

|  |
| --- |
| # ContentPage java 中使用这里的名称  INSERT\_UPDATE ContentPage ;$contentCV[unique=true] ;uid[unique=true] ;name ;masterTemplate(uid,$contentCV) ;label ;defaultPage[default='true'] ;approvalStatus(code)[default='approved'] ;homepage[default='false']  ; ;test01ContentPage ;test 01 ContentPage ;test01PageTemplate ;test01ContentPage ; ; ; |

1. 加入ContentSlotName

|  |
| --- |
| #ContentSlotName  INSERT\_UPDATE ContentSlotName ;compTypeGroup(code) ;name[unique=true] ;owner ;template(uid,$contentCV)[unique=true]  ;wide ;test01-01ContentSlotNameName ; ;test01PageTemplate  ;wide ;test01-02ContentSlotNameName ; ;test01PageTemplate |

1. 设置compoent （里面放置html代码，component有多种类型）

|  |
| --- |
| #Component  INSERT\_UPDATE CMSParagraphComponent ;$contentCV[unique=true] ;uid[unique=true] ;name ;&componentRef  ; ;test01-01Component ;test01-01ComponentName ;test01-01Component  ; ;test01-02Component ;test01-02ComponentName ;test01-02Component  ; ;test01-03Component ;test01-03ComponentName ;test01-03Component |

1. 更新component html内容

|  |
| --- |
| UPDATE CMSParagraphComponent ;$contentCV[unique=true] ;uid[unique=true] ;content[lang=zh]  ; ;test01-01Component ;"<div class=""container mj-Category-nav a1""> div 1 div 1 div 1 </div>"  ; ;test01-02Component ;"<div class=""container mj-Category-nav a2""> div 2 div 2 div 2 </div>"  ; ;test01-03Component ;"<div class=""container mj-Category-nav a3""> div 3 div 3 div 3 </div>" |

7. 加入contentSlot

|  |
| --- |
| #ContentSlot  INSERT\_UPDATE ContentSlot ;$contentCV[unique=true] ;uid[unique=true] ;name ;active ;cmsComponents(&componentRef)  ; ;test01-01ContentSlot ;test01-01ContentSlotName ;true ;test01-01Component  ; ;test01-02ContentSlot ;test01-02ContentSlotName ;true ;test01-02Component,test01-03Component |

8. 关联slot 到template

|  |
| --- |
| #ContentSlotForTemplate  INSERT\_UPDATE ContentSlotForTemplate ;$contentCV[unique=true] ;uid[unique=true] ;position[unique=true] ;pageTemplate(uid,$contentCV)[unique=true][default='test01PageTemplate'] ;contentSlot(uid,$contentCV)[unique=true] ;allowOverwrite  ; ;test01-01ContentSlotForTemplate ;test01-01ContentSlotNameName ; ;test01-01ContentSlot ;true  ; ;test01-02ContentSlotForTemplate ;test01-02ContentSlotNameName ; ;test01-02ContentSlot ;true |

9. 模版页面（jsp页面名称必须与impex中定义的一致：test01Page.jsp）

|  |
| --- |
| <%@ page trimDirectiveWhitespaces=*"true"* %>  <%@ page contentType=*"text/html;charset=UTF-8"* pageEncoding=*"UTF-8"*%>  <%@ taglib prefix=*"template"* tagdir=*"/WEB-INF/tags/desktop/template"* %>  <%@ taglib prefix=*"cms"* uri=*"http://hybris.com/tld/cmstags"* %>  <%@ taglib prefix=*"breadcrumb"* tagdir=*"/WEB-INF/tags/desktop/nav/breadcrumb"* %>  <%@ taglib prefix=*"spring"* uri=*"http://www.springframework.org/tags"* %>  <%@ taglib prefix=*"home"* tagdir=*"/WEB-INF/tags/desktop/home"* %>  <%@ taglib prefix=*"jipin"* tagdir=*"/WEB-INF/tags/desktop/icolor/jiapin"* %>  <%@ taglib prefix=*"c"* uri=*"http://java.sun.com/jsp/jstl/core"*%>  <%@ taglib prefix=*"common"* tagdir=*"/WEB-INF/tags/desktop/common"* %>  <template:page pageTitle=*"*${pageTitle}*"*>  <br>111111<br><br><br>  <cms:pageSlot position=*"test01-01ContentSlotNameName"* var=*"feature"*>  <cms:component component=*"*${feature}*"* element=*"div"*/>  </cms:pageSlot>  <br><br><br><br>2222222222<br>  ===${test01 }====  <br><br>  <cms:pageSlot position=*"test01-02ContentSlotNameName"* var=*"feature"*>  <cms:component component=*"*${feature}*"* element=*"div"*/>  </cms:pageSlot>    <br><br><br>333333<br><br><br>  </template:page> |

10. 访问页面

|  |
| --- |
| @RequestMapping(value = "/test01", method = RequestMethod.***GET***)  **public** String test01(**final** Model model) **throws** CMSItemNotFoundException  {  setUpMetaDataForContentPage(model, getContentPageForLabelOrId("test01ContentPage"));  storeCmsPageInModel(model, getContentPageForLabelOrId("test01ContentPage"));  model.addAttribute("test01", "abcde");  ***LOG***.info("========================================");  **return** getViewForPage(model);  } |

11.cmscockpit访问，修改页面

|  |
| --- |
|  |

在contentSlot中关联component时，一定要注意，被关联的component一定要出现在当前的impex文件中。如：

|  |
| --- |
| UPDATE ContentSlot ;$contentCV[unique=true] ;uid[unique=true] ;cmsComponents(&componentRef)[mode=append]  ; ;test01-01ContentSlot ;TEL01LinkREF,test01-01Component |

在更新test01-01ContentSlot 的时候，TEL01LinkREF，test01-01Component 这两个componoent的定义，一定要出现在文件上方。定义里只需要出现属性&componentRef即可。如：

|  |
| --- |
| UPDATE CMSParagraphComponent ;$contentCV[unique=true] ;uid[unique=true] ;&componentRef  ; ;test01-01Component ;test01-01Component |

如果只是想替换原来的component，直接 把**[mode=append]** 这个去掉

## 34. hybris使用flexible search查询单个属性方法

|  |
| --- |
| **public** String getGiftCardMaxNumber(**final** String numPrefix)  {  **final** FlexibleSearchQuery fQuery = **new** FlexibleSearchQuery(***QUERY\_GIFTCARD\_MAX\_NUMBER***);  fQuery.addQueryParameter("numPrefix", numPrefix);  fQuery.setNeedTotal(**false**);  fQuery.setCount(1);  fQuery.setResultClassList(Collections.*singletonList*(String.**class**));  **final** List<Object> result = getFlexibleSearchService().search(fQuery).getResult();  **if** (CollectionUtils.*isEmpty*(result))  {  **return** **null**;  }  **return** (String) result.get(0);  } |

## 35. hybris查询一定范围内某字段的最大值

前面几位数据是一致的，后面的数字取最大值

|  |
| --- |
| select max({cardNum}) from {giftcard} where substr({cardNum},0,11) = '17021505555' |

## 36 js中操作java数据对象

在一些数据显示时，需要对数据进行处理，或者前台在后续的操作中需要进行过滤，修改数据时，可以通过js来实现。这时候就需要js来操作java传过来的数据。

Java中有如下方法

|  |
| --- |
| @RequestMapping(value = **"/test04"**, method = RequestMethod.***GET***) **public** String test04(**final** Model model)  **throws** CMSItemNotFoundException {  ***LOG***.info(**"--------test04-------"**);  List<OrderData> list = **new** ArrayList<OrderData>();  **for**(**int** i=0;i<10;i++){  OrderData data = **new** OrderData();  data.setCode(**"order-"** + i);  data.setPaymentStatus(**"hello-"** + i\*i);  list.add(data);  }   JSONArray obj = JSONArray.*fromObject*(list);  model.addAttribute(**"list2"**,obj);   setUpMetaDataForContentPage(model, getContentPageForLabelOrId(**"test01ContentPage"**));  storeCmsPageInModel(model, getContentPageForLabelOrId(**"test01ContentPage"**));  **return** getViewForPage(model); } |

页面显示数据方式可以使用如下

|  |
| --- |
| <c:forEach var="data" items="${list2}">  <br>  ====${data.code}=== </c:forEach> <hr> |

Js 中使用java数据如下

|  |
| --- |
| <**script type="text/javascript"**>  **var *x*** = **'${**list2**}'**;  **var *obj*** = eval(***x***);  **console**.**info**(**"=====xxxxxxxxxxx==========="** + ***obj***);  *$*(**function**(){   **for**(**var** i=0;i<***obj***.**length**;i++){  **console**.**info**(**"AAAAAA----"** + ***obj***[i].**code**);  }  });  </**script**> |

## 37 java生成验证码

1. jsp页面

|  |
| --- |
| captcha:<**img id="captchaImg" src="./test05"**/> |

1. js 重新加载页面（需要jquery）

|  |
| --- |
| **$img** = *$*(**"#captchaImg"**); **$img**.on(**"click"**,**function**(){  *$*(**this**).attr(**"src"**,**"./test05?r="** +***Math***.random()); }); |

1. java中的方法

|  |
| --- |
| @RequestMapping(value = **"test05"**, method = RequestMethod.***GET***) **public void** test05(**final** Model model, HttpServletResponse response)  **throws** Exception {  ***LOG***.info(**"--------test05-------"**);  response.setHeader(**"Pragma"**, **"No-cache"**);  response.setHeader(**"Cache-Control"**, **"no-cache"**);  response.setDateHeader(**"Expires"**, 0);  response.setContentType(**"image/jpeg"**);  *//生成随机字串* **final** String verifyCode = VerifyCodeUtils.*generateVerifyCode*(4);  *//生成图片* **final int** w = 120, h = 50;  VerifyCodeUtils.*outputImage*(w, h, response.getOutputStream(), verifyCode); } |

工具类

|  |
| --- |
| */\*\*  \*  \*/* **package** com.npz.icolor.storefront.util;  **import** java.awt.Color; **import** java.awt.Font; **import** java.awt.Graphics; **import** java.awt.Graphics2D; **import** java.awt.RenderingHints; **import** java.awt.geom.AffineTransform; **import** java.awt.image.BufferedImage; **import** java.io.File; **import** java.io.FileOutputStream; **import** java.io.IOException; **import** java.io.OutputStream; **import** java.util.Arrays; **import** java.util.Random;  **import** javax.imageio.ImageIO;   */\*\*  \** ***@author*** *jh  \*  \*/* **public class** VerifyCodeUtils {  *//使用到Algerian字体，系统里没有的话需要安装字体，字体只显示大写，去掉了1,0,i,o几个容易混淆的字符* **public static final** String ***VERIFY\_CODES*** = **"23456789ABCDEFGHJKLMNPQRSTUVWXYZ"**;  **private static** Random *random* = **new** Random();    */\*\*  \* 使用系统默认字符源生成验证码  \*  \** ***@param verifySize*** *\* 验证码长度  \** ***@return*** *\*/* **public static** String generateVerifyCode(**final int** verifySize)  {  **return** *generateVerifyCode*(verifySize, ***VERIFY\_CODES***);  }   */\*\*  \* 使用指定源生成验证码  \*  \** ***@param verifySize*** *\* 验证码长度  \** ***@param sources*** *\* 验证码字符源  \** ***@return*** *\*/* **public static** String generateVerifyCode(**final int** verifySize, String sources)  {  **if** (sources == **null** || sources.length() == 0)  {  sources = ***VERIFY\_CODES***;  }  **final int** codesLen = sources.length();  **final** Random rand = **new** Random(System.*currentTimeMillis*());  **final** StringBuilder verifyCode = **new** StringBuilder(verifySize);  **for** (**int** i = 0; i < verifySize; i++)  {  verifyCode.append(sources.charAt(rand.nextInt(codesLen - 1)));  }  **return** verifyCode.toString();  }   */\*\*  \* 生成随机验证码文件,并返回验证码值  \*  \** ***@param w*** *\** ***@param h*** *\** ***@param outputFile*** *\** ***@param verifySize*** *\** ***@return*** *\** ***@throws*** *IOException  \*/* **public static** String outputVerifyImage(**final int** w, **final int** h, **final** File outputFile, **final int** verifySize)  **throws** IOException  {  **final** String verifyCode = *generateVerifyCode*(verifySize);  *outputImage*(w, h, outputFile, verifyCode);  **return** verifyCode;  }   */\*\*  \* 输出随机验证码图片流,并返回验证码值  \*  \** ***@param w*** *\** ***@param h*** *\** ***@param os*** *\** ***@param verifySize*** *\** ***@return*** *\** ***@throws*** *IOException  \*/* **public static** String outputVerifyImage(**final int** w, **final int** h, **final** OutputStream os, **final int** verifySize)  **throws** IOException  {  **final** String verifyCode = *generateVerifyCode*(verifySize);  *outputImage*(w, h, os, verifyCode);  **return** verifyCode;  }   */\*\*  \* 生成指定验证码图像文件  \*  \** ***@param w*** *\** ***@param h*** *\** ***@param outputFile*** *\** ***@param code*** *\** ***@throws*** *IOException  \*/* **public static void** outputImage(**final int** w, **final int** h, **final** File outputFile, **final** String code) **throws** IOException  {  **if** (outputFile == **null**)  {  **return**;  }  **final** File dir = outputFile.getParentFile();  **if** (!dir.exists())  {  dir.mkdirs();  }  **try** {  outputFile.createNewFile();  **final** FileOutputStream fos = **new** FileOutputStream(outputFile);  *outputImage*(w, h, fos, code);  fos.close();  }  **catch** (**final** IOException e)  {  **throw** e;  }  }   */\*\*  \* 输出指定验证码图片流  \*  \** ***@param w*** *\** ***@param h*** *\** ***@param os*** *\** ***@param code*** *\** ***@throws*** *IOException  \*/* **public static void** outputImage(**final int** w, **final int** h, **final** OutputStream os, **final** String code) **throws** IOException  {  **final int** verifySize = code.length();  **final** BufferedImage image = **new** BufferedImage(w, h, BufferedImage.***TYPE\_INT\_RGB***);  **final** Random rand = **new** Random();  **final** Graphics2D g2 = image.createGraphics();  g2.setRenderingHint(RenderingHints.***KEY\_ANTIALIASING***, RenderingHints.***VALUE\_ANTIALIAS\_ON***);  **final** Color[] colors = **new** Color[5];  **final** Color[] colorSpaces = **new** Color[]  { Color.***WHITE***, Color.***CYAN***, Color.***GRAY***, Color.***LIGHT\_GRAY***, Color.***MAGENTA***, Color.***ORANGE***, Color.***PINK***, Color.***YELLOW*** };  **final float**[] fractions = **new float**[colors.**length**];  **for** (**int** i = 0; i < colors.**length**; i++)  {  colors[i] = colorSpaces[rand.nextInt(colorSpaces.**length**)];  fractions[i] = rand.nextFloat();  }  Arrays.*sort*(fractions);   g2.setColor(Color.***GRAY***);*// 设置边框色* g2.fillRect(0, 0, w, h);   **final** Color c = *getRandColor*(200, 250);  g2.setColor(c);*// 设置背景色* g2.fillRect(0, 2, w, h - 4);   *//绘制干扰线* **final** Random random = **new** Random();  g2.setColor(*getRandColor*(160, 200));*// 设置线条的颜色* **for** (**int** i = 0; i < 20; i++)  {  **final int** x = random.nextInt(w - 1);  **final int** y = random.nextInt(h - 1);  **final int** xl = random.nextInt(6) + 1;  **final int** yl = random.nextInt(12) + 1;  g2.drawLine(x, y, x + xl + 40, y + yl + 20);  }   *// 添加噪点* **final float** yawpRate = 0.05f;*// 噪声率* **final int** area = (**int**) (yawpRate \* w \* h);  **for** (**int** i = 0; i < area; i++)  {  **final int** x = random.nextInt(w);  **final int** y = random.nextInt(h);  **final int** rgb = *getRandomIntColor*();  image.setRGB(x, y, rgb);  }   *shear*(g2, w, h, c);*// 使图片扭曲* g2.setColor(*getRandColor*(100, 160));  **final int** fontSize = h - 4;  **final** Font font = **new** Font(**"Algerian"**, Font.***ITALIC***, fontSize);  g2.setFont(font);  **final char**[] chars = code.toCharArray();  **for** (**int** i = 0; i < verifySize; i++)  {  **final** AffineTransform affine = **new** AffineTransform();  affine.setToRotation(Math.***PI*** / 4 \* rand.nextDouble() \* (rand.nextBoolean() ? 1 : -1),  (w / verifySize) \* i + fontSize / 2, h / 2);  g2.setTransform(affine);  g2.drawChars(chars, i, 1, ((w - 10) / verifySize) \* i + 5, h / 2 + fontSize / 2 - 10);  }   g2.dispose();  ImageIO.*write*(image, **"jpg"**, os);  }   **private static** Color getRandColor(**int** fc, **int** bc)  {  **if** (fc > 255)  {  fc = 255;  }  **if** (bc > 255)  {  bc = 255;  }  **final int** r = fc + *random*.nextInt(bc - fc);  **final int** g = fc + *random*.nextInt(bc - fc);  **final int** b = fc + *random*.nextInt(bc - fc);  **return new** Color(r, g, b);  }   **private static int** getRandomIntColor()  {  **final int**[] rgb = *getRandomRgb*();  **int** color = 0;  **for** (**final int** c : rgb)  {  color = color << 8;  color = color | c;  }  **return** color;  }   **private static int**[] getRandomRgb()  {  **final int**[] rgb = **new int**[3];  **for** (**int** i = 0; i < 3; i++)  {  rgb[i] = *random*.nextInt(255);  }  **return** rgb;  }   **private static void** shear(**final** Graphics g, **final int** w1, **final int** h1, **final** Color color)  {  *shearX*(g, w1, h1, color);  *shearY*(g, w1, h1, color);  }   **private static void** shearX(**final** Graphics g, **final int** w1, **final int** h1, **final** Color color)  {   **final int** period = *random*.nextInt(2);   **final boolean** borderGap = **true**;  **final int** frames = 1;  **final int** phase = *random*.nextInt(2);   **for** (**int** i = 0; i < h1; i++)  {  **final double** d = (period >> 1) \* Math.*sin*((**double**) i / (**double**) period + (6.2831853071795862D \* phase) / frames);  g.copyArea(0, i, w1, 1, (**int**) d, 0);  **if** (borderGap)  {  g.setColor(color);  g.drawLine((**int**) d, i, 0, i);  g.drawLine((**int**) d + w1, i, w1, i);  }  }   }   **private static void** shearY(**final** Graphics g, **final int** w1, **final int** h1, **final** Color color)  {   **final int** period = *random*.nextInt(40) + 10; *// 50;* **final boolean** borderGap = **true**;  **final int** frames = 20;  **final int** phase = 7;  **for** (**int** i = 0; i < w1; i++)  {  **final double** d = (period >> 1) \* Math.*sin*((**double**) i / (**double**) period + (6.2831853071795862D \* phase) / frames);  g.copyArea(i, 0, 1, h1, 0, (**int**) d);  **if** (borderGap)  {  g.setColor(color);  g.drawLine(i, (**int**) d, i, 0);  g.drawLine(i, (**int**) d + h1, i, h1);  }   }   }   **public static void** main(**final** String[] args) **throws** IOException  {  **final** File dir = **new** File(**"F:/verifies"**);  **final int** w = 200, h = 80;  **for** (**int** i = 0; i < 50; i++)  {  **final** String verifyCode = *generateVerifyCode*(4);  **final** File file = **new** File(dir, verifyCode + **".jpg"**);  *outputImage*(w, h, file, verifyCode);  }  } } |

## 37 mysql自动增长ID 起始值

|  |
| --- |
| ALTER TABLE 表名 AUTO\_INCREMENT=10000000; |

## 38 使用cglib生成代理对象

1. 使用jar包：



如果没有导入asm.jar会错 org.objectweb.asm.Type这个类找不到。

1. 实例

未实现接口，未使用final的类

|  |
| --- |
| **package** com.tim.proxy.cglib;  **public** **class** BookServiceImpl  {  **public** **void** addBook(String book)  {  System.***out***.println("====add book======" + book);  }  } |

产生代理对象的类

|  |
| --- |
| **package** com.tim.proxy.cglib;  **import** java.lang.reflect.Method;  **import** net.sf.cglib.proxy.Enhancer;  **import** net.sf.cglib.proxy.MethodInterceptor;  **import** net.sf.cglib.proxy.MethodProxy;  **public** **class** BookServiceImplCglib **implements** MethodInterceptor  {  **private** Object target;  **public** Object getInstance(Object target)  {  **this**.target = target;  Enhancer enhancer = **new** Enhancer();  enhancer.setSuperclass(**this**.target.getClass());  // 回调方法  enhancer.setCallback(**this**);  // 创建代理对象  **return** enhancer.create();  }  @Override  **public** Object intercept(Object obj, Method method, Object[] aobj, MethodProxy methodproxy) **throws** Throwable  {  System.***out***.println("add book start...");  method.invoke(target, aobj);  System.***out***.println("add book end...");  **return** **null**;  }  } |

测试结果类

|  |
| --- |
| **package** com.tim.proxy.cglib;  **public** **class** Main  {  **public** **static** **void** main(String[] args)  {  BookServiceImplCglib bookServiceCglib = **new** BookServiceImplCglib();  BookServiceImpl cglib = (BookServiceImpl) bookServiceCglib.getInstance(**new** BookServiceImpl());  cglib.addBook("Thinking java");  }  } |

## 39 mysql保存数据时，数据内容太大

使用mysql保存数据时，如果 字段内容太大，会发生以下错误：

|  |
| --- |
| Packet for query is too large (8742 > 1024) |

这个是由于mysql设置的缓存太小了5.7默认只有4194304b = 4M。

我们需要在mysql中重新覆盖该数据,执行以下sql脚本：

|  |
| --- |
| SET GLOBAL max\_allowed\_packet =100\*1024\*1024 |

可以将缓存设置为100M，这样在存数据时，就不会有问题了。

查询缓存大小的sql如下：

|  |
| --- |
| SHOW VARIABLES LIKE '%max\_allowed\_packet%' |

## 40. springMVC 使用

1. web.xml文件中，加入dispatcherServlet，对整个项目的Controller控制

|  |
| --- |
| <?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\_2\_5.xsd"* id=*"WebApp\_ID"* version=*"2.5"*>  <display-name>spring0400</display-name>  <welcome-file-list>  <welcome-file>index.html</welcome-file>  <welcome-file>index.htm</welcome-file>  <welcome-file>index.jsp</welcome-file>  <welcome-file>default.html</welcome-file>  <welcome-file>default.htm</welcome-file>  <welcome-file>default.jsp</welcome-file>  </welcome-file-list>    <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:applicationContext.xml</param-value>  </context-param>    <servlet>  <servlet-name>springmvc</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <load-on-startup>1</load-on-startup>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:springmvc-servlet.xml</param-value>  </init-param>  </servlet>    <servlet-mapping>  <servlet-name>springmvc</servlet-name>  <url-pattern>/</url-pattern>  </servlet-mapping>    </web-app> |

**注**：springmvc中，urlPattern的值 ，一定是 【 / 】,不能是【/\*】

1. springmvc需要一个管理controller层的xml文件。默认会去WEB-INF找springmvc-servlet.xml这个文件。其中，红色的部分是在web.xml里DispatcherServlet对应的名称。

我们在web.xml中已经指定了从classpath里找这个文件，所以，不会再从web-inf里找。

|  |
| --- |
| <?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:context=*"http://www.springframework.org/schema/context"*  xmlns:mvc=*"http://www.springframework.org/schema/mvc"* xmlns:aop=*"http://www.springframework.org/schema/aop"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context.xsd*  *http://www.springframework.org/schema/aop*  *http://www.springframework.org/schema/aop/spring-aop.xsd*  *http://www.springframework.org/schema/mvc*  *http://www.springframework.org/schema/mvc/spring-mvc.xsd"*>  <context:component-scan base-package=*"com.tim.spring0400.mvc"* />  <mvc:annotation-driven/>    <mvc:interceptors>  <ref bean=*"helloInterceptor"*/>  </mvc:interceptors>      <bean id=*"jspViewResolver"*  class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>    <!-- 加这个，就要加入jstl.jar -->  <!-- <property name="viewClass"  value="org.springframework.web.servlet.view.JstlView" /> -->    <property name=*"prefix"* value=*"/WEB-INF/views/"* />  <property name=*"suffix"* value=*".jsp"* />  </bean>    </beans> |

1. spring中有自己的xml文件，我们也定义在了classpath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"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:mvc=*"http://www.springframework.org/schema/mvc"*  xmlns:aop=*"http://www.springframework.org/schema/aop"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context.xsd*  *http://www.springframework.org/schema/aop*  *http://www.springframework.org/schema/aop/spring-aop.xsd*  *http://www.springframework.org/schema/mvc*  *http://www.springframework.org/schema/mvc/spring-mvc.xsd"*  default-autowire=*"byType"*>  <!-- <bean id="helloService" class="com.tim.spring0400.service.impl.HelloServiceImpl"></bean> -->  <context:component-scan base-package=*"com.tim.spring0400"*/>  <aop:aspectj-autoproxy/>    </beans> |

1. Controller写法

|  |
| --- |
| **package** com.tim.spring0400.mvc;  **import** org.springframework.stereotype.Controller;  **import** org.springframework.ui.Model;  **import** org.springframework.validation.BindingResult;  **import** org.springframework.web.bind.annotation.ModelAttribute;  **import** org.springframework.web.bind.annotation.RequestMapping;  **import** org.springframework.web.bind.annotation.RequestMethod;  **import** org.springframework.web.servlet.ModelAndView;  **import** com.tim.spring0400.model.Student;  @Controller  @RequestMapping("/hello")  **public** **class** HelloController  {  } |

* 1. 方法写法

|  |
| --- |
| @RequestMapping(method = RequestMethod.***GET***)  **public** String sayHello(Model model)  {  model.addAttribute("message", "world");  return new ModelAndView("/WEB-INF/views/home.jsp"); } |

注: 在springmvc-servlet.xml中如果没有定义**jspViewResolver ，** 需要在返回的时候，写上页面所在的长路径。但是如果配置了视图解析器，则只要写上页面的名称即可。这里用到了常说的约定优于配置。

方法中，有个参数Model，这个可以做为返回数据的容器。把要返回的数据，直接放进行，在jsp页面，通过${message}就可以接收。

这个方法名上没有写ResquestMapping的映射，所以在调用时，直接 <http://localhost:8080/spring0400/hello> 就可以访问。 /hello是Controller对应的映射。

|  |
| --- |
| @RequestMapping(value = "/home3", method = RequestMethod.***GET***)  **public** String sayHello3(Model model)  {  Student stu = **new** Student();  model.addAttribute("stu", stu);  **return** "input";  } |

注：这个方法可以把一个对象设置到jsp页面上去。在页面中，做为表单的commandName。

|  |
| --- |
| @RequestMapping(value = "/home3", method = RequestMethod.***POST***)  **public** String sayHello4(@ModelAttribute("stu") Student stu, BindingResult result, Model model)  {  System.***out***.println(stu.getId() + " " + stu.getName());  **new** StudentValidate().validate(stu, result);  **if** (result.hasErrors())  {  model.addAttribute("stu", stu);  **return** "input";  }  model.addAttribute("stu", stu);  **return** "home";  } |

注：在jsp页面中，如果使用了spring的form表单，在这里可以直接通过Student stu 这个参数进行接收。其中参数名stu要与页面上的 commandName一致。

这个方法可以对参数进行正确性校验，spring的Validator接口，可以实现这个功能。我们实现这个接口，对参数进行合法性校验。如果参数有问题，则可以返回到提交的页面，也就是 return “input”.这个页面。如果想实现这个功能，需要在Student stu这个参数之前加上 @ModelAttribute("stu") 这个一定要与commandName一致。然后，在这个参数后面紧跟着接收错误信息的参数BindingResult result，这样，在返回页面后，就可以获取到错误信息提示了。

Jsp页面 。 input.jsp

|  |
| --- |
| <%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*  pageEncoding=*"ISO-8859-1"*%>  <%@ taglib prefix=*"s"* uri=*"http://www.springframework.org/tags/form"*%>  <%@ taglib prefix=*"c"* uri=*"http://java.sun.com/jsp/jstl/core"*%>  <!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>  student<hr>  <c:url var=*"url"* value=*"/hello/home3"*/>  <s:form commandName=*"stu"* action=*"*${url }*"* method=*"post"*>  id:<s:input path=*"id"*/><br>  <font color=*"red"*><s:errors path=*"id"* /></font><br>  name:<s:input path=*"name"*/><br>  <font color=*"red"*><s:errors path=*"name"* /></font><br>  <s:button value=*"submit"*>submit</s:button>  </s:form>  </body>  </html> |

这其中用到了spring form标签，jstl标签（servlet-api.jar） 。

错误信息放在<s:errors path=*"id"* />就可以接收了。

1. Validator实现

|  |
| --- |
| package com.tim.spring0400.mvc;  import org.springframework.validation.Errors;  import org.springframework.validation.Validator;  import com.tim.spring0400.model.Student;  public class StudentValidate implements Validator  {  @Override  public boolean supports(Class<?> arg0)  {  return Student.class.equals(arg0);  }  @Override  public void validate(Object arg0, Errors error)  {  Student s = (Student) arg0;  if (s.getId() == 2)  {  error.rejectValue("id", "error.num", "defaultValueID");  }  if (s.getName().length() < 3)  {  error.rejectValue("name", "name length error", "defaultValueName");  }  }  } |

Git 代码<https://github.com/Hhtianwei/spring0400.git>

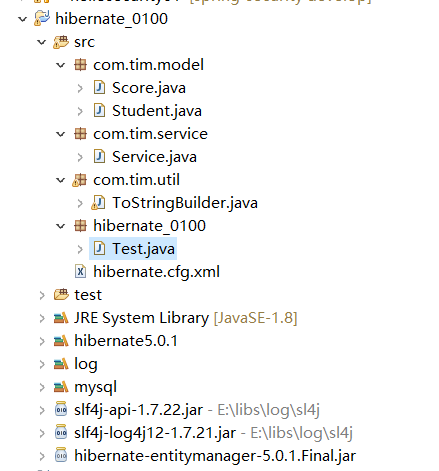
## 41. springMVC （spring security,hibernate）集成

Git位置：

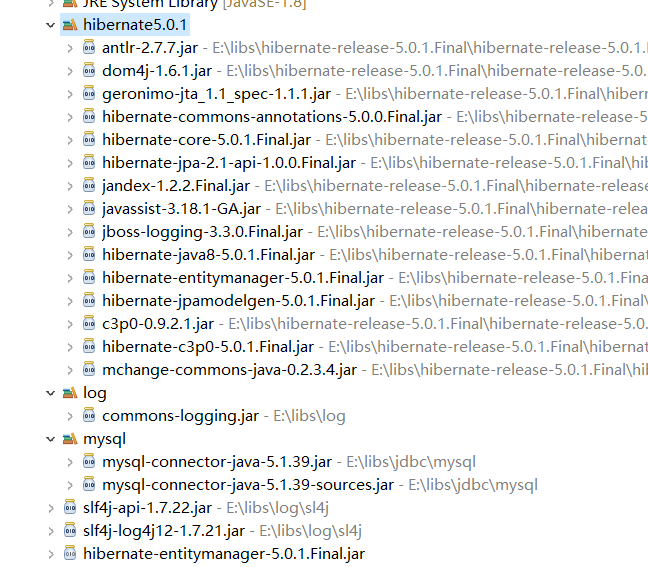
|  |
| --- |
| https://github.com/Hhtianwei/springMVC\_model\_0100.git |

## 42.hibernate 5.0使用

1. 项目结构：



1. 使用jar:



1. Hibernate.cfg.xml配置

|  |
| --- |
| <?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>    <property name=*"connection.driver\_class"*>com.mysql.jdbc.Driver</property>  <property name=*"connection.url"*>jdbc:mysql://localhost/mytest2</property>  <property name=*"connection.username"*>root</property>  <property name=*"connection.password"*>root</property>  <property name=*"dialect"*>org.hibernate.dialect.MySQL5Dialect</property>  <!-- JDBC connection pool (use the built-in) -->  <property name=*"connection.pool\_size"*>1</property>  <!-- Enable Hibernate's automatic session context management -->  <property name=*"current\_session\_context\_class"*>thread</property>  <!-- Disable the second-level cache -->  <!-- Echo all executed SQL to stdout -->  <property name=*"show\_sql"*>true</property>  <property name=*"format\_sql"*>true</property>  <property name=*"hbm2ddl.auto"*>update</property>  <mapping class=*"com.tim.model.Student"*/>  <mapping class=*"com.tim.model.Score"*/>  </session-factory>  </hibernate-configuration> |

1. 创建SessionFactory

|  |
| --- |
| StandardServiceRegistryBuilder builder = **new** StandardServiceRegistryBuilder();  builder.configure();  StandardServiceRegistry registry = builder.build();  MetadataSources metadataSources = **new** MetadataSources(registry);  MetadataBuilder metadataBuilder = metadataSources.getMetadataBuilder();  Metadata metadata = metadataBuilder.build();  SessionFactory factory = metadata.buildSessionFactory();  Session session = factory.openSession();  session.beginTransaction();  //Student s2 = session.load(Student.class, 2);  //System.out.println(s2.toString());  //Score score1 = session.load(Score.class, 2);  //Thread.sleep(2000);  //int eng = score1.getEnglish();  //Student stu = score1.getStu();  //System.out.println(eng + "--------" + stu.getBirthday());  //sc.setStu(s);  //session.save(s);  session.save(sc);  session.getTransaction().commit();  session.close();  factory.close(); |

1. Model创建

|  |
| --- |
| package com.tim.model;  import java.util.Date;  import javax.persistence.Column;  import javax.persistence.Entity;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import javax.persistence.Table;  import com.tim.util.ToStringBuilder;  @Entity  @Table(name = "student")  public class Student  {  private int id;  private String name;  private Date birthday;  @Id  @Column(name = "id", unique = true, nullable = false)  @GeneratedValue(strategy = GenerationType.IDENTITY)  public int getId()  {  return id;  }  public void setId(int id)  {  this.id = id;  }  @Column(name = "name", unique = false, nullable = false)  public String getName()  {  return name;  }  public void setName(String name)  {  this.name = name;  }  @Column(name = "birthday", unique = false, nullable = true)  public Date getBirthday()  {  return birthday;  }  public void setBirthday(Date birthday)  {  this.birthday = birthday;  }  @Override  public String toString()  {  return ToStringBuilder.reflectionToString(this);  }  } |

* 1. 级联model

类Score ,里面包含了一个student对象

|  |
| --- |
| package com.tim.model;  import java.util.Date;  import javax.persistence.CascadeType;  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.OneToOne;  import javax.persistence.Table;  @Entity  @Table(name = "score")  public class Score  {  private int id;  private int english;  private Date testDate;  private Student stu;  @Id  @Column(name = "id", nullable = false, unique = true)  @GeneratedValue(strategy = GenerationType.IDENTITY)  public int getId()  {  return id;  }  public void setId(int id)  {  this.id = id;  }  @Column(name = "english", nullable = true, unique = false)  public int getEnglish()  {  return english;  }  public void setEnglish(int english)  {  this.english = english;  }  @Column(name = "testDate", nullable = true, unique = false)  public Date getTestDate()  {  return testDate;  }  public void setTestDate(Date testDate)  {  this.testDate = testDate;  }  @OneToOne(cascade = CascadeType.ALL)  @JoinColumn(name = "studentid")  public Student getStu()  {  return stu;  }  public void setStu(Student stu)  {  this.stu = stu;  }  } |

在score中的stu属性的get方法加上 @OneToOne,@JoinColumn标签。（这是一个单向一对一关联）

数据库里保存的是student的id,但是我们在关联的时候，使用对象。

## 43. servlet3.0 上传文件

Servlet:

|  |
| --- |
| **package** com.tim.fileupload.servlet;  **import** java.io.File;  **import** java.io.IOException;  **import** javax.servlet.ServletException;  **import** javax.servlet.annotation.MultipartConfig;  **import** javax.servlet.annotation.WebServlet;  **import** javax.servlet.http.HttpServlet;  **import** javax.servlet.http.HttpServletRequest;  **import** javax.servlet.http.HttpServletResponse;  **import** javax.servlet.http.Part;  @WebServlet(urlPatterns = "/upload")  @MultipartConfig  **public** **class** FileUploadServlet **extends** HttpServlet  {  @Override  **protected** **void** doPost(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException  {  Part file = req.getPart("fileName");  String fileName = file.getSubmittedFileName();  String newPath = "D://upload//";//如果选择项目内的一个目录，会在tomcat重启之后清空文件，所以应该选择一个服务器所在机器的硬盘上的一个空间  file.write(newPath + File.***separator*** + fileName);  req.getRequestDispatcher("/").forward(req, resp);  }  @Override  **protected** **void** doGet(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException  {  **this**.doPost(req, resp);  }  } |

Jsp

|  |
| --- |
| <%@ page language=*"java"* contentType=*"text/html; charset=utf-8"*  pageEncoding=*"utf-8"*%>  <%  String path = request.getContextPath();  String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";  %>  <!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=utf-8"*>  <title>Insert title here</title>  </head>  <body>  index.jsp <br>  上传文件功能:<br>  <form action=*"*<%=path %>*/upload"* method=*"post"* enctype=*"multipart/form-data"*>  <input type=*"file"* name=*"fileName"*/>  <input type=*"submit"* value=*"上传"*/>  </form>  </body>  </html> |

## 44. spring bean 标签的parent

在spring bean中定义一个抽象bean:

|  |
| --- |
| <bean id=*"absractUserBean"* abstract=*"true"*>  <property name=*"name"* value=*"hello01"*></property>  <property name=*"service"* ref=*"studentService"*></property>  </bean> |

因为是抽象的，所以可以不用对应到具体的类中。

定义子类去继承这个抽象的bean

|  |
| --- |
| <bean id=*"testAbstractBean"* class=*"com.tim.spring.test.TestAbstractBean"* parent=*"absractUserBean"*>  <property name=*"id"* value=*"123"*/>  </bean> |

因为子类 bean 继承了父类的bean,所以子类bean中也会有父类的属性[name,service],而且要求，子类中，一定要有对应的 name, service 属性。父类的引用会直接用到他们身上。

## 45. Java web中spring管理下的filter

Web.xml中定义filter。该filter是通过Spring的代理Filter进行管理

|  |
| --- |
| <!-- log record -->  <filter>  <filter-name>icolorAdminLogFilter</filter-name>  <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>  <init-param>  <param-name>targetBeanName</param-name>  <param-value>logRecordFilter</param-value>  </init-param>  </filter>  <filter-mapping>  <filter-name>icolorAdminLogFilter</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping> |

注： targetBeanName 这个参数就是在spring bean中进行管理的Filter类。

Beans.xml

|  |
| --- |
| <bean id=*"logRecordFilter"* class=*"com.nippon.icolor.base.filter.IcolorAdminLogFilter"* init-method=*"initParams"*>  <property name=*"rememberLogService"* ref=*"icolorAdminLogService"*/>  <property name=*"checkAllURL"* value=*"false"*/>  </bean> |

Bean中引用了spring中的其它serivce。是可以被注入进去的。

Java：

|  |
| --- |
| **public** **class** IcolorAdminLogFilter **implements** Filter  {  **private** IcolorAdminLogService rememberLogService;  @Override  **public** **void** doFilter(**final** ServletRequest servletRequest, **final** ServletResponse servletResponse, **final** FilterChain filterChain)  **throws** IOException, ServletException  {  doSomething(S);  filterChain.doFilter(servletRequest, servletResponse);  }  rememberLogService,set,get  } |

## 46. html页面引用css，js加参数

为防止html页面使用缓存中的css,js而导致最新的样式文件没有被使用。通过给引用的css,js文件加入版本号的参数来强制刷新页面中的文件。

|  |
| --- |
| <link rel="stylesheet" class="delayLoad" data-href="../styles/index.min.css">  <script class="delayLoad" data-src="../scripts/common.js"></script>  <script>  (function () {  var delayTag = document.querySelectorAll(".delayLoad"),  len = delayTag.length,  version = "1.0.0"; //版本号调整  for (var i = 0; i < len; i++) {  (function (idx) {  var target, attr;  if (delayTag[idx].getAttribute("data-href")) {  target = delayTag[idx].getAttribute("data-href");  attr = "href";  } else {  target = delayTag[idx].getAttribute("data-src");  attr = "src";  }  delayTag[idx][attr] = target + "?v=" + version;  })(i);  }  }());  </script> |

结果如图：

