# crm-01

1.搭建CRM项目的开发环境：

\* 设置字体

\* 设置工作区的字符集为utf-8

\* 设置basePath变量及base标签

<%

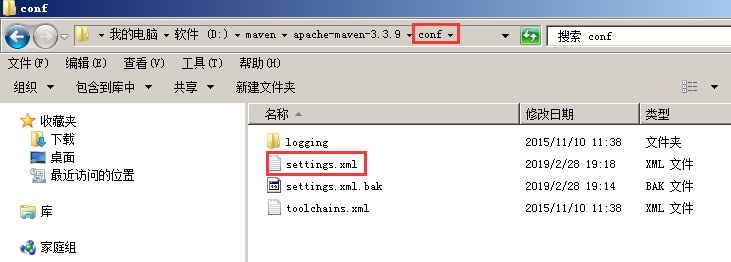
String basePath = request.getScheme() + "://" + request.getServerName() + ":" + request.getServerPort() + request.getContextPath() + "/";

%>

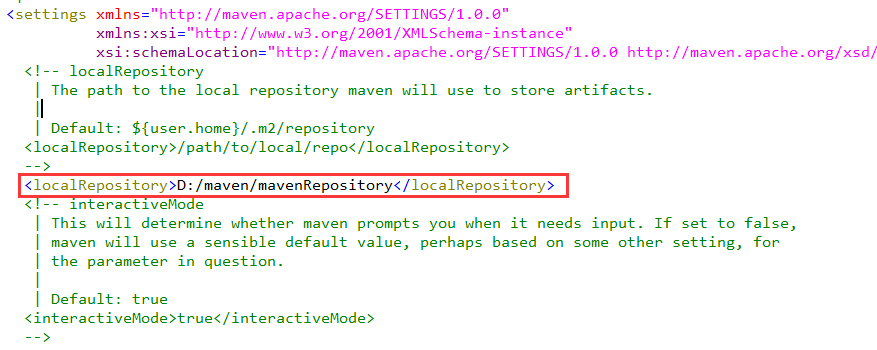
<base href="<%=basePath%>">

创建Maven项目前的准备工作

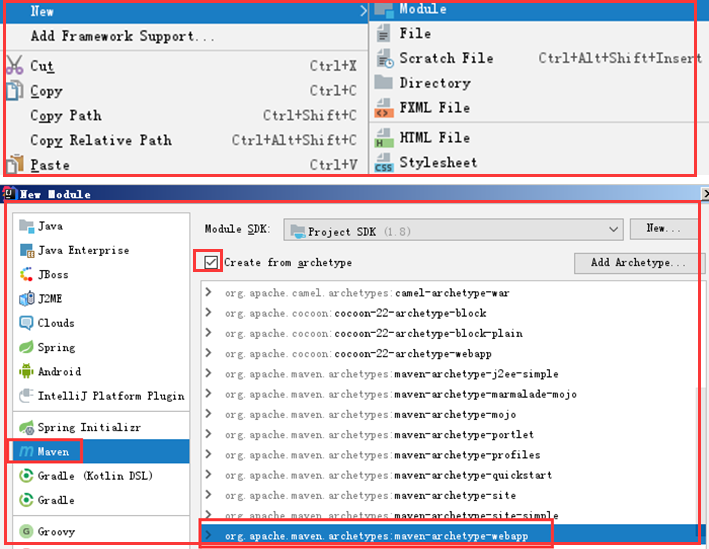
使用settings引入本地仓库

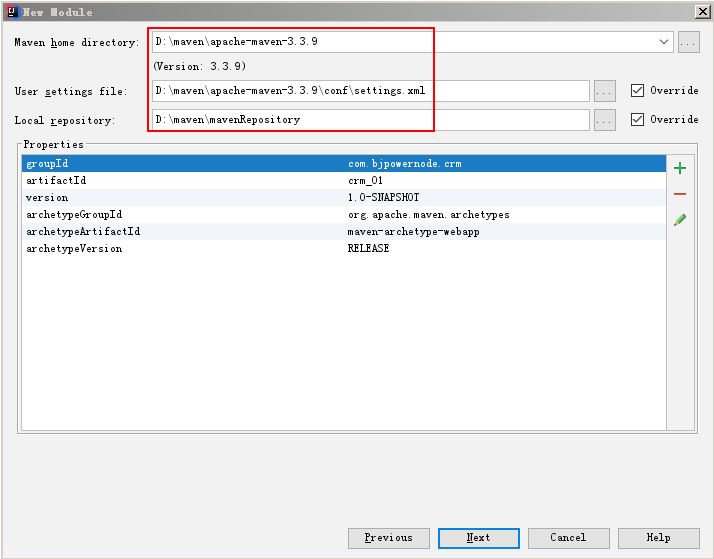


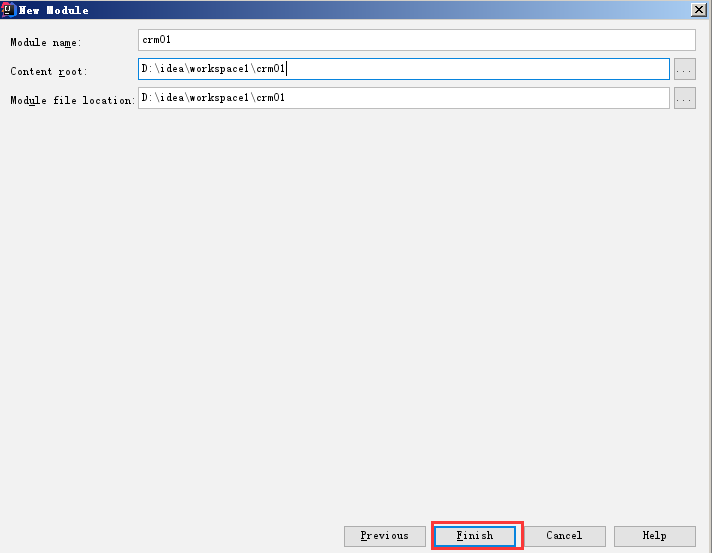
打开settings.xml设置本地仓库位置



创建Maven web项目







新创建的Maven项目，目录结构如下：



将pom.xml中没有用的信息去除掉

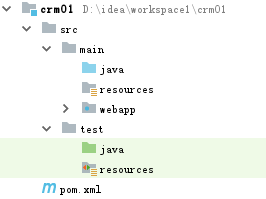


在main文件夹下建立java文件夹和resources文件夹

与main文件夹平级创建一个test文件夹

test文件夹下创建java文件夹和resources文件夹

**注意：为文件夹赋予功能（颜色）后，文件夹才会生效**

****

将我们自己的web.xml模板导入

通过pom.xml导入我们需要的jar包



在pom.xml中加入构建资源

<build>

<resources>

<resource>

<directory>src/main/java</directory><!--所在的目录-->

<includes><!--包括目录下的.properties,.xml文件都会扫描到-->

<include>\*\*/\*.properties</include>

<include>\*\*/\*.xml</include>

</includes>

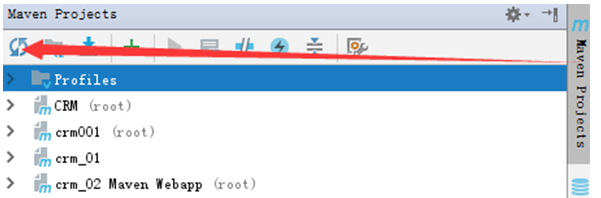
<filtering>false</filtering>

</resource>

</resources>

</build>

点击左侧Maven Projects，点击刷新



在resource路径下引入开发用的配置文件和属性文件

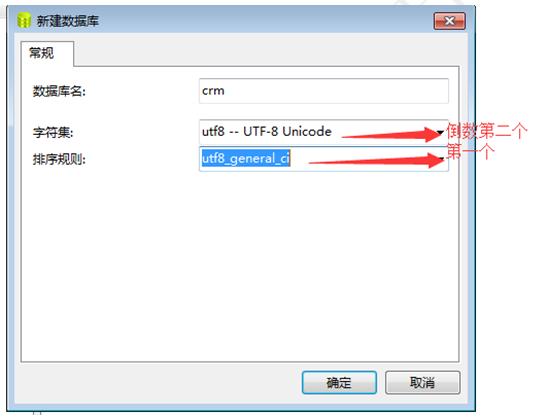
jdbc.properties（修改数据库名）

log4j.properties

mybatis-config.xml

SqlMapper.xml（建立后台包结构后加入该文件）

\* 创建一个crm数据库

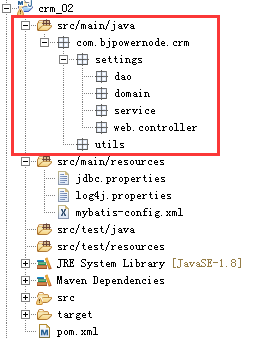


\* 将crm原型拷贝到webapp目录下

\* 启动Tomcat服务器，测试

# crm-02

1、创建项目后端结构（结构标准化说明）



\* 提供一些相关的工具类

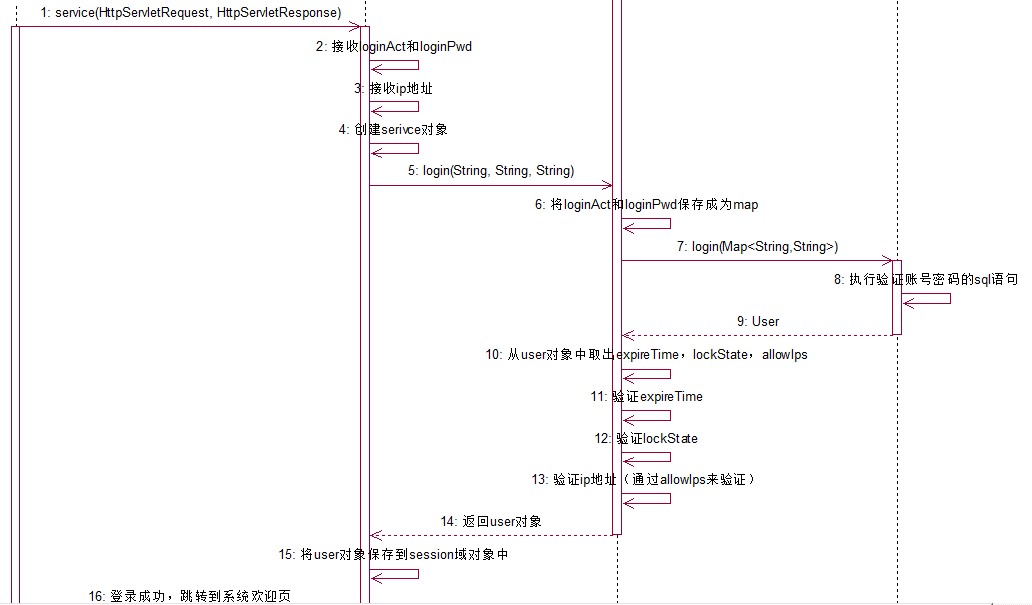
SqlSessionUtil等..

\* 搭建用户相关的domain、dao、service、controller

\* 对于dao层命名的说明

# crm-03

时序图：用来描绘从浏览器到服务器执行全流程的结构图



对于url-pattern

/abc/myServlet.do

/abc/\*

/\*

\*.do

**/abc/\*.do 错误**

**abc/\*.do 错误**

用户登录

1、创建用户表tbl\_user

\*导入用户表登录基础数据

\*对于MD5的说明

2、用户登录需求

\* login.html修改为login.jsp，解决404.

\* 页面加载完毕之后，“用户名”自动获得焦点。

关键代码：

$("#loginAct").focus();

\* 用户点击登录可以登录，用户回车键也可以登录。

关键代码：

// 点击登录按钮，则登录。

$("#loginBtn").click(function(){

login();

});

//点击回车也登录

$(window).keydown(function(event){

// event只是一个变量名，随意编写，是一个引用，指向了一个事件对象，所 有的键盘事件对象都有一个keyCode属性，用来获取键值。

if(event.keyCode == 13){ // 回车键是13

login();

}

});

\* 登录采用ajax post

\* 登录时用户名和密码不能为空

2、使用异常处理机制来完成错误信息提示

创建exception包结构，创建自定义异常继承Exception

关键代码：

publicLoginException(String msg){

super(msg);

}

3、登录条件

验证用户名密码正确

验证失效时间

关键代码：

compareTo...

验证锁定状态

验证ip地址

关键代码：

contains...

登录成功后跳转到workbench/index.jsp

在工作台初始页展现用户名

# crm-04

1、登录验证的过滤器。

所有用户必须登录之后才能访问该web资源。

拦截什么？

\*.do

\*.jsp

不拦截什么？

/login.jsp

/login.do

2、登录成功之后，在右上角显示用户登录的信息。

3、使login.jsp始终在顶层窗口中打开

关键代码：

if(window.top!=window){

window.top.location=window.location;

}

# crm-05

1、市场活动表的设计。

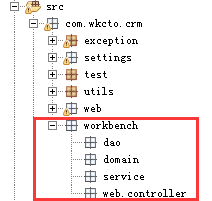
建表：tbl\_activity

tbl\_activity\_remark

2、市场活动相关页面全部修改为jsp,解决404错误。

3、创建工作台包结构，搭建内部基本组件

**更新mybatis主配置文件**



4、展现新增市场活动modal窗口：

\* 所有者必须动态

5、怎么让下拉列表默认选中某一项？

$("#grade").val("2");

<select id="grade">

<option value="1">高中</option>

<option value="2">专科</option>

<option value="3">本科</option>

</select>

6、发送ajax post请求保存市场活动。

# crm-06

日历控件

保存市场活动的时候，需要填写日期，一般情况下都是需要相关的日历控件的。

我们这里使用的前端UI框架是基于Bootstrap实现的，为了统一风格，所有的插件建议使用Bootstrap扩展的插件。

我们这里使用的是：bootstrap datetimepicker（日期拾取器）

引用bootstrap datetimepicker插件的步骤：

（1）引入jQuery的库

（2）引入bootstrap的库

（3）引入datetimepicker的css库和js库（注意引入先后顺序）

<link rel="stylesheet" type="text/css" href="jquery/bootstrap-datetimepicker-master/css/bootstrap-datetimepicker.min.css">

<script type="text/javascript" src="jquery/bootstrap-datetimepicker-master/js/bootstrap-datetimepicker.js"></script>

<script type="text/javascript" src="jquery/bootstrap-datetimepicker-master/locale/bootstrap-datetimepicker.zh-CN.js"></script>

（4）设置time类属性

$(".time").datetimepicker({

language: "zh-CN",

format: "yyyy-mm-ddhh:ii:ss",//显示格式

minView: "hour",//设置只显示到月份

initialDate: new Date(),//初始化当前日期

autoclose: true,//选中自动关闭

todayBtn: true, //显示今日按钮

clearBtn : true,

pickerPosition: "bottom-left"

});

$(".time").datetimepicker({

minView: "month",

language: 'zh-CN',

format: 'yyyy-mm-dd',

autoclose: true,

todayBtn: true,

pickerPosition: "bottom-left"

});

（5）在相关组件的class下引入time类标识

例如：



# crm-07

实现分页查询市场活动：

后端处理：

1、给分页查询专门封装一个VO对象。(关于VO类的说明)

关键代码：

public class PaginationVO<T> {

privateint total;

private List<T>dataList;

{“total”:100,”dataList”:[{},{},{}]}

当前版本只实现服务端，不管前端。

2、使用PaginationVO对象，响应JSON

注意：owner为用户名（user.name）



前端测试搭建：市场活动列表页面加载完毕之后，分页查询，显示第一页数据。

$(function(){

display(1 , 2);

});

function display(pageNo , pageSize){

...

}

测试路径参考：

http://127.0.0.1:8080/crm/workbench/activity/page.do?pageNo=1&pageSize=2&owner=%E4%B8%89&name=2&startDate=2018-07-16

# crm-08

实现分页查询市场活动：

前端处理：

1、动态展现列表数据

2、使用pagination分页插件

（1）导入pagination插件相关css和js

<link rel="stylesheet" type="text/css" href="jquery/bs\_pagination/jquery.bs\_pagination.min.css">

<script type="text/javascript" src="jquery/bs\_pagination/jquery.bs\_pagination.min.js"></script>

<script type="text/javascript" src="jquery/bs\_pagination/en.js"></script>

（2）在列表下面导入分页组件div（将原有默认分页div去掉）

<div id="activityPage"></div>

（3）**在pageList.do处理ajax返回值后**，加入分页组件

$("#activityPage").bs\_pagination({

currentPage: pageNo, // 页码

rowsPerPage: pageSize, // 每页显示的记录条数

maxRowsPerPage: 20, // 每页最多显示的记录条数

totalPages: totalPages, // 总页数

totalRows: data.total, // 总记录条数

visiblePageLinks: 3, // 显示几个卡片

showGoToPage: true,

showRowsPerPage: true,

showRowsInfo: true,

showRowsDefaultInfo: true,

onChangePage : function(event, data){

pageList(data.currentPage , data.rowsPerPage);

}

});

处理总页数totalPages

（4）调用分页插件

例如点击查询按钮（**注意：查询按钮需要设施type="button"**）

$("#searchBtn").click(function(){

pageList(1 ,$("#activityPage").bs\_pagination('getOption', 'rowsPerPage'));

});

（5）js中对于totalPages的处理

vartotalPages = data.total%pageSize==0 ? data.total/pageSize :parseInt(data.total/pageSize)+1;

（6）ajax对于时间戳的加入

cache : false

2、分页问题处理

问题1：

在查询框中输入内容，不点击查询按钮

点击分页按钮

结果为查询框中的内容生效了

问题2：

在查询框中输入内容，点击查询按钮

再在查询框中输入内容，不点击查询按钮

点击分页按钮

结果为新的查询框中的内容生效了

2、翻页的时候，查询条件是如何处理的？

将查询条件放到隐藏域当中，每一次翻页的时候，条件都从隐藏域当中取。

什么时候更新隐藏域？

（1）点击查询按钮的时候将查询框中的内容更新（保存内容到）隐藏域。

hidden-->search

（2）执行pageList的时候，将隐藏域中的内容更新熬查询框。

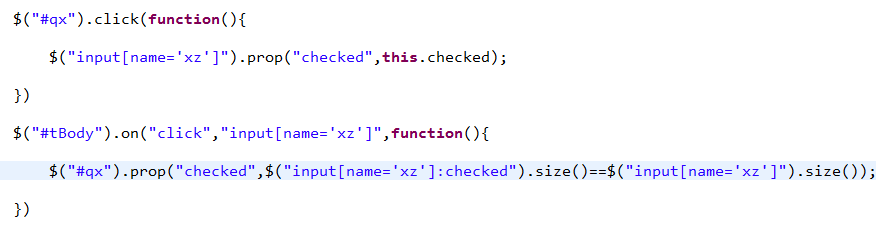
search-->hidden

3、复选框的全选和取消全选。

使用jQuery实现。

重点1：jQuery支持多个元素一块处理。很多地方是不需要使用each遍历的。

重点2：为后期ajax动态生成的元素绑定事件，必须用on。



\*其他模块的分页查询独立完成多加练习

# crm-09

1、删除市场活动：

\* 删除的时候至少选中一条记录。

\* 删除之前要提示用户是否确定要删除（删除数据是比较危险的动作，必须提示！）

\* 一次可以删除多条记录。

\* 删除的时候发送ajax post请求。

\* 删除成功之后显示第一页数据。

\* 删除市场活动的关联删除市场活动的备注。

\* URL：

/crm/workbench/activity/deleteById.do?id=xx&id=xx&id=xx&id=xx....

# crm-10

修改市场活动：

注意：打开修改的modal窗口的过程需要activity对象和userList对象  
 （打开添加的modal窗口，仅需要userList对象）

（1）弹出修改市场活动的modal窗口：

\* 一次只能修改一个。

\* 点击修改的时候，发送ajax get请求，提交市场活动id，根据id获取市场活动信息，展示modal。

（2）注意修改editBy和editTime

（3）更新后，在哪一页还回到哪一页

关键代码

pageList($("#activityPage").bs\_pagination('getOption', 'currentPage')

,$("#activityPage").bs\_pagination('getOption', 'rowsPerPage'));

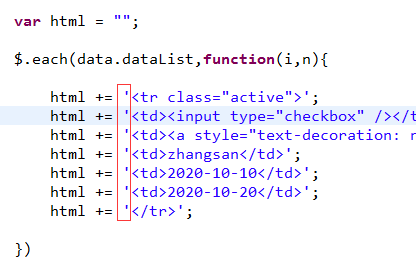
# crm-11

1、查看市场活动明细

\* 使用传统请求全局刷新完成跳转到详细信息页的操作

\* 只展示市场活动的信息。（注意：市场活动的信息中所有者以用户名的形式存在，需要表连接。）

注意：查询详细信息的超链接是有json动态响应生成的，所以需要在拼接前端结构的时候使用单引号进行拼接。



关键代码：

html += '<td><a style="text-decoration: none; cursor: pointer;" onclick="window.location.href=\'workbench/activity/detail.do?id='+n.id+'\';">'+n.name+'</a></td>';

# crm-12

1、市场活动明细页面加载完毕之后发送ajax get请求，取出该市场活动下所有的备注信息。

2、生成的备注信息，以before的形式添加在备注输入框的上方。

关键代码：

$("#remarkDiv").before(html);

3、使用editFlag来操控显示的创建/修改时间创建/修改人

关键代码：

html += '<font color="gray">市场活动</font><font color="gray">-</font><b>${activity.name}</b><small style="color: gray;">'

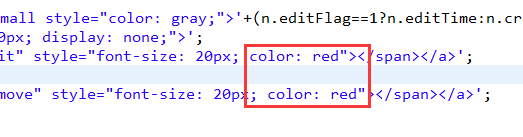
+(n.editFlag==0?n.createTime:n.editTime)+'由'+(n.editFlag==0?n.createBy:n.editBy)+'</small>';

**注意1：能不能这么用？**

**注意2：before的使用方式**

**注意3：(n.editFlag == 0 ?n.createTime : n.editTime)的判断方式**

4、将动态显示的备注信息的修改和删除图标修改为不变红色。



5、使用 on操控动态生成的备注的修改和删除按钮，鼠标划入划出操作。

关键代码：



$("#remarkBody").on("mouseover",".remarkDiv",function(){

$(this).children("div").children("div").show();

})

$("#remarkBody").on("mouseout",".remarkDiv",function(){

$(this).children("div").children("div").hide();

})

# crm-13

删除备注

1、设置被删除的div的id为备注的id。

关键代码



2、删除信息直接在动态生成信息的删除图片超链接上触发事件

关键代码



3、注意：使用该形式触发事件的方法需要写在$(function(){})的外面

4、在方法中使用ajax异步删除。

5、删除成功后，需要将该备注信息相关div组件移除

关键代码

if(data.success){

$("#"+id).remove();

}else{

alert("删除失败");

}

# crm-14

添加备注

1、添加前验证备注信息不能为空。

2、发送ajax post请求添加备注。

3、添加备注成功之后前端页面动态添加一个div

4、添加备注成功之后清空用户填写的备注信息。

# crm-15

修改备注：

1、点击修改按钮之后，弹出修改备注的modal窗口。

2、修改备注的modal窗口上显示备注信息。

修改窗口的初始值需要内容和id（注意搭配<h5 id="\_备注id">）

关键代码：

varnoteContent = $.trim($("#\_"+id).html());

**注意：**

**和**

3、用户点击更新，修改备注：

- 备注内容不能为空。

- 更新的时候发送ajax post请求。

- 更新成功之后，隐藏modal窗口。

- 更新前端的div。

关键代码：



# crm-16

1、导入“线索”、“客户”、“联系人”、“交易”相关表

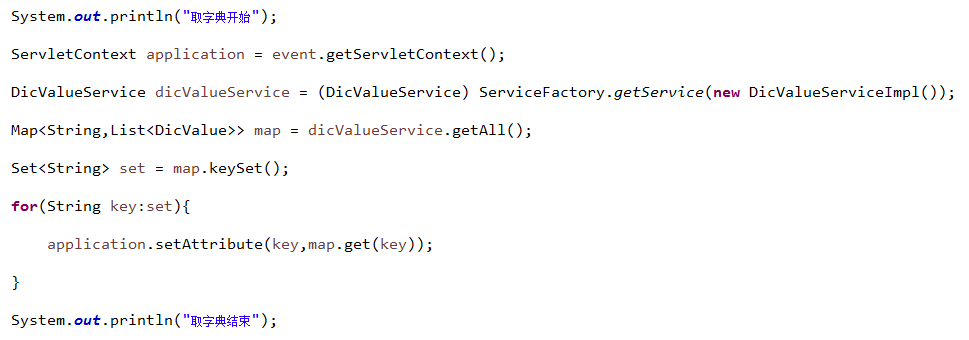
2、将“线索”、“客户”、“联系人”、“交易”模块的html修改为jsp，解决404错误。

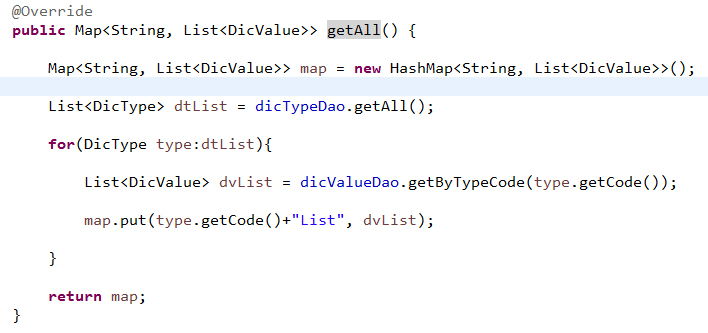
3、搭建“线索”、“客户”、“联系人”、“交易”相关后端结构(domain,dao,service,controller)

4、解析crm中的多对多关系

5、cache（缓存机制）解决数据字典存储问题

关键代码





# crm-17

创建线索

1、创建线索相关domain，dao，service，controller

2、点击“创建”弹出创建线索的modal窗口：

\* 所有者动态

将当前登录用户设置为默认所有者

$("#owner").val("${user.id}");

\* 称呼、线索状态、线索来源需要从application域中取出。

称呼：appellationList

线索状态：clueStateList

线索来源：sourceList

3、保存线索

表单验证

ajax post

# crm-18

查看线索的详细信息

关联用户表owner

后台使用request域搭配转发即可

由于演示没有做列表所以暂时将id写固定值即可



# crm-19

1、线索明细页面加载完毕之后，发送ajax get请求取出该线索关联的所有市场活动，展现活动列表。

/crm/workbench/clue/getActivityByClueId.do?clueId=xxxx

三张表连接：

tbl\_activity a

tbl\_clue\_activity\_relation car

tbl\_user u

select

a.name,a.startDate,a.endDate,u.name as owner,car.id as relationId

from

tbl\_activity a

join

tbl\_clue\_activity\_relation car

on

a.id = car.activityId

join

tbl\_user u

on

a.owner = u.id

where

car.clueId = #{clueId}

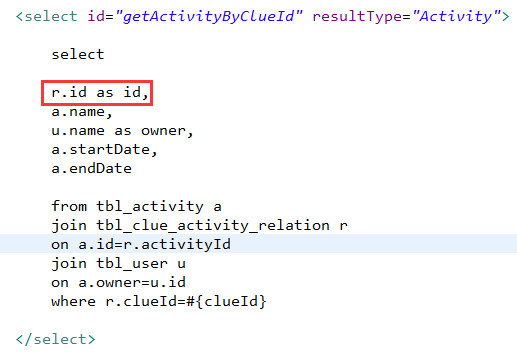
# crm-20

解除关联（解除市场活动和线索的关联关系。）

（1）在市场活动后点击解除关联

（2）触发onclick事件，使用隐藏域保存关联表id的同时，打开解除市场活动的 模态窗口

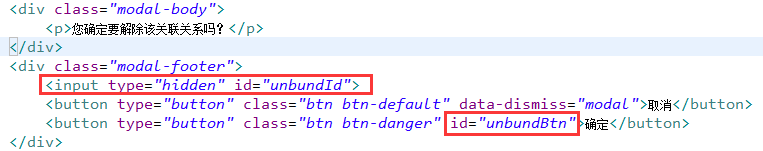
**注意将n.id设置为关联表id**

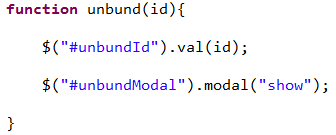
****

**注意id的传值方式必须加引号引起来**



**将id设置到隐藏域**





**点击确定按钮，发出ajax请求解除关联，传递关联表id即可**

（3）将关联id传到后台，通过删除关联表数据，解除关联关系

（4）解除关联后，调用刷新市场活动方法。

# crm-21

1、弹出"关联市场活动"modal窗口:

\* modal窗口弹出之前：清空tbody

\* modal窗口弹出之前：清空输入框

\* modal窗口弹出之前：重置复选框

prop("checked":false);

\* 弹出modal窗口的时候不查询所有的市场活动，直到用户输入查询条件并且捕 捉到键值13的时候，再查询。

关键代码：

$("#chkId").keydown(function(event){

if(event.keyCode==13){

2、用户输入查询条件，然后回车键，根据市场活动的名称进行查询。

Bootstrap的modal窗口只要触发回车键，modal窗口会自动关闭！

关键代码

return false;

3、注意子查询,查询时剔除掉已经关联的市场活动

<select id="getActivityByName" resultType="Activity">

select

a.id,

a.name,

a.startDate,

a.endDate,

u.name as owner

fromtbl\_activity a

jointbl\_user u

on a.owner=u.id

where a.name like '%' #{name} '%'

and a.id not in(

selectactivityId

fromtbl\_clue\_activity\_relation

whereclueId=#{clueId}

)

</select>

# crm-22

1、detail.jsp页面跳转到Convert.jsp页面：

detail.jsp向convert.jsp页面传数据：

detail.jsp?company=xxx&fullname=xxxx&appellation=xxx....

detail.jsp传给convert.jsp数据，在convert.jsp页面怎么取数据呢？

使用EL表达式来取数据，在EL表达式当中有一个隐含对象param，可以直接使用。

语法格式：

${param.name}

2、用户在转换页面点击转换按钮转换线索：

分析：转换的实现步骤？

(1) 获取到线索id，通过线索id获取线索对象（线索对象当中封装了线索的信息）

(2) 通过线索对象提取客户信息，当该客户不存在的时候，新建客户（根据公司的名称精确匹配，判断该客户是否存在！）

(3) 通过线索对象提取联系人信息，保存联系人

(4) 线索备注转换到客户备注以及联系人备注

(5) “线索和市场活动”的关系转换到“联系人和市场活动”的关系

(6) 如果有创建交易需求，创建一条交易

(7) 如果创建了交易，则创建一条该交易下的交易历史

(8) 删除线索备注

(9) 删除线索和市场活动的关系

(10) 删除线索

# crm-23

跳转到交易添加页

1、添加页要求：

\* 所有者动态

\* 预计成交日期日历控件

\* 阶段动态

\* 类型动态

\* 来源动态

\* 市场活动源和联系人名称固定id(预备数据)

\* 下次联系时间日历控件



2、EL表达式支持三目运算符：

${user.id equserObj.id ? "selected" : ""}

EL表达式当中他们有什么区别？

${user} ${"user"}的区别是什么？

${user} 表示从四个域对象当中取出user对象，输出到浏览器。

${"user"} 表示把普通的字符串"user"输出到浏览器。

3、选择不同的阶段，动态生成不同的可能性

\* change事件

\* 添加一个配置文件：该文件中配置了“阶段”对应的“可能性”。

\* 属性文件中只能存储unicode编码，不能直接存储中文，高版本的eclipse会自动将中文转换成unicode码。

以前，我们开发的时候，eclipse还没有这个功能，那个时候我们通常会手动的借助JDK自带的工具进行unicode码

转换：这个工具的名字叫做：native2ascii.exe

\* 在服务器启动阶段，解析Stage2Possibility.properties属性文件，将解析的数据存放到Map集合当中，

后端关键代码：

Map集合存储到application域当中。

ResourceBundle bundle = ResourceBundle.getBundle("com.wkcto.crm.resources.Stage2Possibility");

Enumeration<String> keys2 = bundle.getKeys();

Map<String,String>possibilityMap = new HashMap<>();

while(keys2.hasMoreElements()){

String key = keys2.nextElement();

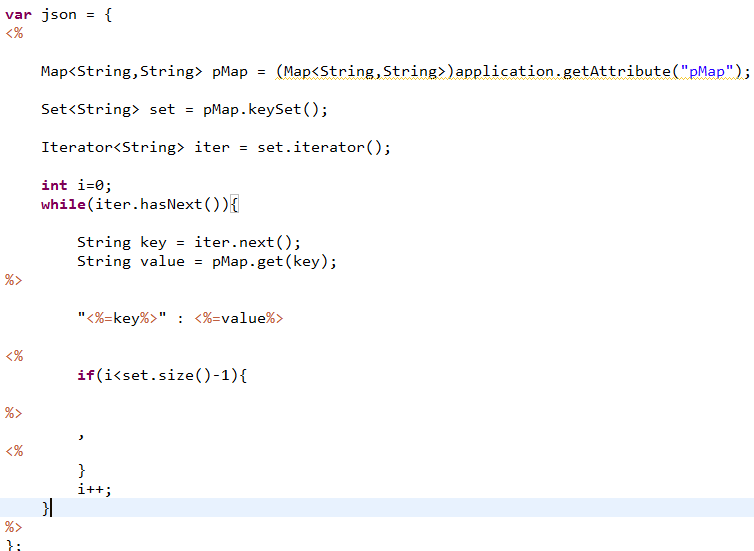
String value = bundle.getString(key);

possibilityMap.put(key, value);

}

application.setAttribute("pMap", possibilityMap);

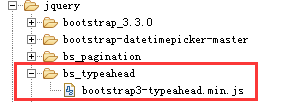
前端关键代码：



4、客户名称支持自动补全

使用bootstrap typeahead插件

（1）加入typeahead插件并引入到jsp中





（2）使用方式

关键代码：

$("#create-customerName").typeahead({

source: function (query, process) {

$.post(

"workbench/transaction/getCustomerName.do",

{ "name" : query },

function (data) {

//alert(data);

process(data);

},

"json"

);

},

delay: 1500

});

注意：

（1）后台查客户名字即可List<String>

（2）将浏览器默认提示关闭autocomplete="off"

# crm-24

1、保存交易

（1）添加页点击按钮提交表单,正常submit即可

（2）提交表单客户上传名字，后台处理，最终保存的是customerId

（3）查询是否有此客户，如果没有则添加客户记录

（4）添加交易记录（注意先后顺序，添加完客户之后，才知道客户id）

（5）添加交易历史

（6）添加后重定向到列表页

# crm-25

1、查看交易的明细：



tbl\_tran t 交易表

tbl\_user u 用户表

tbl\_customer c 客户表

tbl\_activity a 市场活动表

tbl\_contacts con 联系人表

select

t.id,u.name as owner,t.money,t.name,t.expectedDate,c.name as customerId,t.stage,

t.type,t.source,a.name as activityId,con.fullname as contactsId,t.createBy,t.createTime,

t.editBy,t.editTime,t.description,t.contactSummary,t.nextContactTime

from

tbl\_tran t

join

tbl\_user u

on

t.owner = u.id

join

tbl\_customer c

on

t.customerId = c.id

left join

tbl\_activity a

on

t.activityId = a.id

left join

tbl\_contacts con

on

t.contactsId = con.id

where

t.id = ?;

# crm-26

1、交易明细页面展示之后，发送ajax请求，展示出该交易的“阶段历史”。

2、在交易历史domain中加入新属性possibility，用来显示可能性。

# crm-27

详细信息页**动态展示阶段图标**

图标展示：

****表示已完成的阶段class="glyphiconglyphicon-ok-circle mystage"

style="color: #90F790;"

****表示进行中的阶段class="glyphiconglyphicon-map-marker mystage"

style="color: #90F790;"

****表示未完成的阶段class="glyphiconglyphicon-record mystage"

表示交易失败class="glyphiconglyphicon-remove mystage"

表示已该形式交易失败class="glyphiconglyphicon-remove mystage" style="color: red;"

跳转到详细信息页之后：

（1）准备工作

在阶段展示图标区取得当前阶段，通过pMap取得可能性

从上下文中取得stageList

（2）判断可能性

**if(当前阶段可能性为0){**

//遍历数据字典值集合stageList，将所有的阶段取出

for(DicValuedv:dvList){

if(遍历出来的阶段可能性为0){

if(如果是当前阶段){

//展现图标为

//class="glyphiconglyphicon-remove mystage"

//style="color: red;"

}else{ //如果不是当前阶段

//class="glyphiconglyphicon-remove mystage"

}

}else {//遍历出来的阶段可能性不为0

//class="glyphiconglyphicon-record mystage"

}

}

**}else{ //可能性不为0**

intcurrentIndex = 0;

//遍历数据字典值集合stageList，将所有的阶段取出

//如果遍历出来的阶段为当前阶段，则为currentIndex变量赋值i

//注意：及时终止循环提高效率

//遍历数据字典值集合stageList，将所有的阶段取出

for(DicValuedv:dvList){

if(阶段可能性为0){

//class="glyphiconglyphicon-remove mystage"

}else{//阶段可能性不为0

if(i ==currentIndex){ //根据i判断如果是当前阶段

//表示正在进行

//class="glyphiconglyphicon-map-marker mystage"

//style="color: #90F790;"

}else if(i<currentIndex){

//表示已完成

//class="glyphiconglyphicon-ok-circle mystage"

//style="color: #90F790;"

}else{

//表示未完成

//class="glyphiconglyphicon-record mystage"

}

}

}

}

注意：鼠标停留展现信息方式为data-content="<%=dv.getText()%>"

最后展示预计成交日期

# crm-28

1、用户点击图标，修改交易的阶段信息（这个版本先不更新图标，只更新内容）：

发送ajax请求：

上传参数：

tranId

stage

money（创建交易历史用）

expectedDate（创建交易历史用）

修改交易操作：

更新阶段

更新修改者

更新修改时间

添加阶段历史

更新成功后：

页面局部刷新阶段、可能性、修改者、修改时间

# crm-29

更新“交易阶段成功”之后，更新每一个图标。

交易中的状态



丢失后的状态1



丢失后的状态2



1、为图标span标签加入id，值为当前循环下标i

2、点击图标触发onclick事件，除了将状态参数传入，同时将当前下标 i变量传入。

关键代码：

onclick="changeStage('<%=dv.getValue()%>','<%=i%>')"

3、更改交易状态后，触发更新交易状态图标样式的方法，该方法接收上述i变量 （index）。

4、创建int count

5、从上下文中取得pMap，遍历pMap统计可能性不为0的阶段的数量，统计为 count

6、更新图标开始

//将count转换为js变量point

//point当做圈和叉的分界点

//index为当前更新图标下标

if(当前更新图标下标大于等于分界点){ //说明是后面的叉，不是圈

//将i设置为分界点

for(var i = point; i <<%=pMap.size()%>; i++){//循环所有叉，排除圈

if(i等于当前更新图标的下标){

//将当前图标（叉）变红

//关键代码：$("#" + i).css("color" , "red");

}else{//i不是当前更新图标的下标

//将当前图标（叉）变黑

}

}

//遍历分界点前面的所有图标（遍历所有圈，排除叉）

for(var i = 0; i < point; i++){

//移除原来所有样式removeClass

//为分界点前所有元素赋予圈图标addClass

//赋予黑色css

}

}else{//说明是前面的圈，不是叉

//遍历分界点前面所有的圈

for(var i = 0; i < point; i++){

if(i为当前更新图标下标){

//移除所有样式removeClass

//赋予正在进行的图标addClass

//glyphiconglyphicon-map-marker mystage

//赋予绿色#90F790

}else if(i小于当前更新图标下标){//已经完成的圈

//移除所有样式removeClass

//赋予已完成的图标addClass

//glyphiconglyphicon-ok-circle mystage

//赋予绿色#90F790

}else{//还没有完成的圈

//移除所有样式removeClass

//赋予未完成的图标addClass

//glyphiconglyphicon-record mystage

//赋予黑色

}

}

//从分界点遍历后面所有的叉

for(var i = point; i <<%=pMap.size()%>; i++){

//赋予黑色

}

}

# crm-30

统计：

统计图表-->交易统计图表

将超链接改为jsp

workbench/chart/transaction/index.jsp

根据官方提供案例搭建基础漏斗图

修改标题

title: {

text: '交易漏斗图',

subtext: '不同阶段交易数量统计'

}

toolbox选择性去掉

legend去掉

calculable去掉

series属性设置：

name:'交易漏斗图',

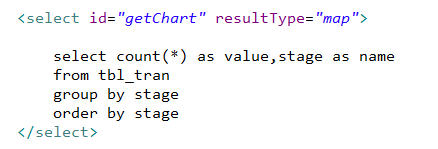
max: data.total,

data: data.dataList

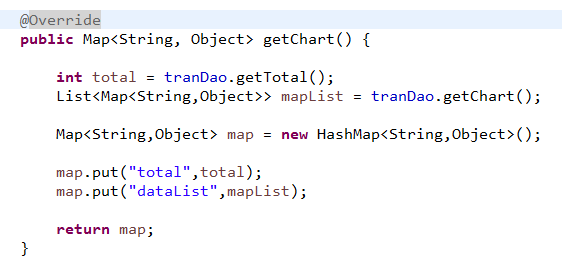
发出ajax请求workbench/transaction/getChart.do，取得data数据

数据格式：{value: count(\*), name: stage}

使用分组函数取得数据



打包数据：



返回数据格式

