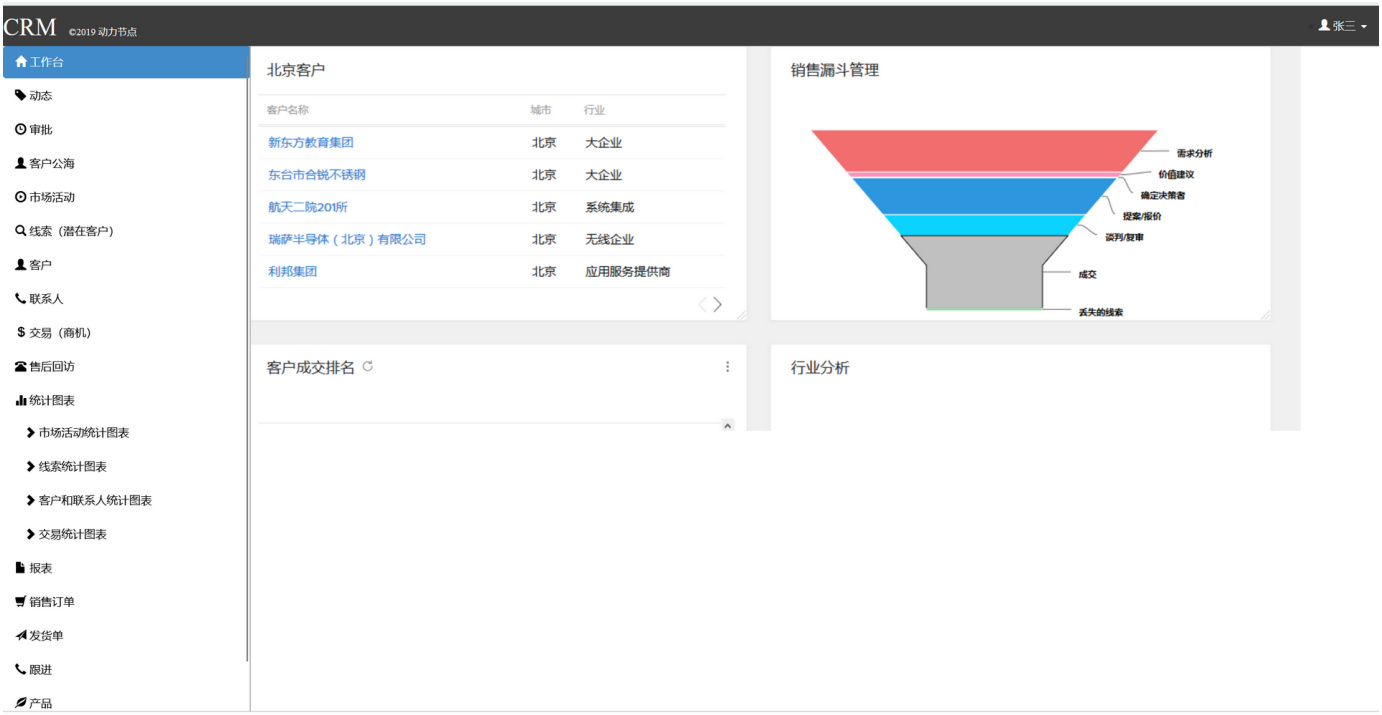
7创建市场活动

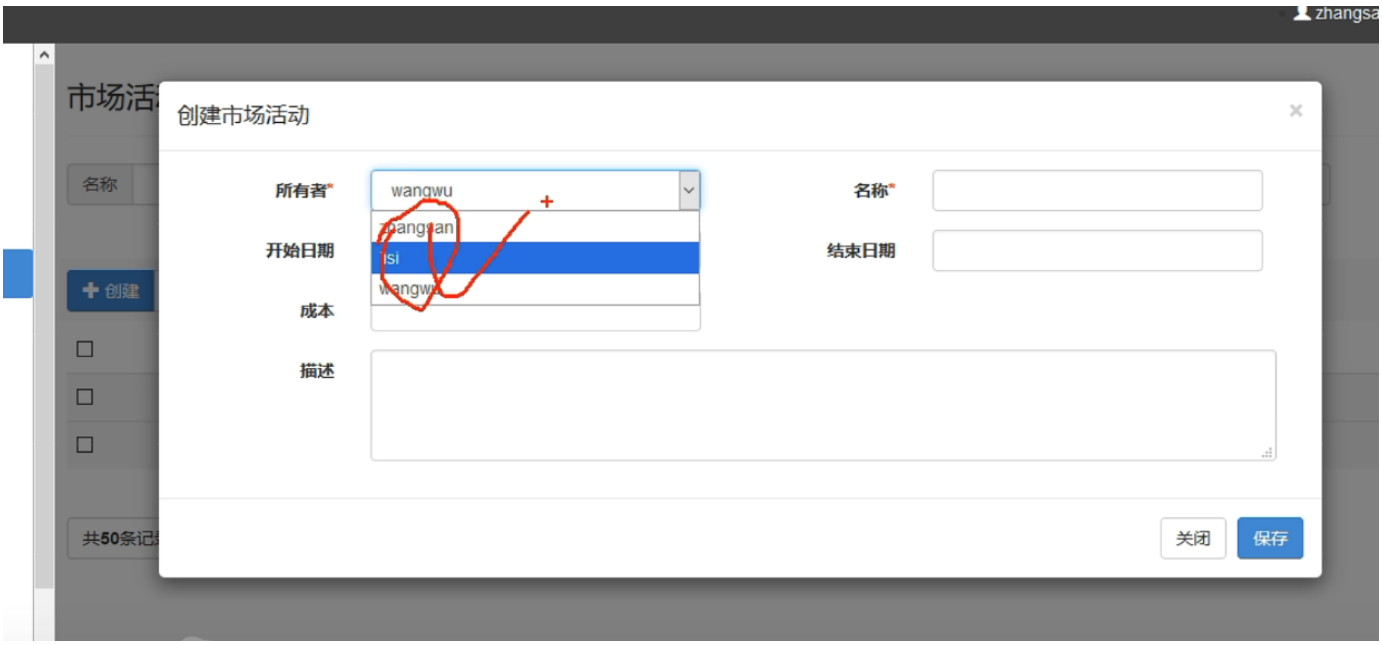


- 用户在市场活动主页面,点击"创建"按钮,弹出创建市场活动的模态窗口;
- 用户在创建市场活动的模态窗口填写表单,点击"保存"按钮,完成创建市场活动的功能.
- *所有者是动态的 (在现实市场活动主页面时，就从数据库中查询出所有用户并且显示在创建的模态窗口中)
- *所有者和名称不能为空（这个前台怎么实现，前台怎么提交）
- *如果开始日期和结束日期都不为空,则结束日期不能比开始日期小
- *成本只能为非负整数
- *创建成功之后,关闭模态窗口,刷新市场活动列，显示第一页数据，保持每页显示条数不变
- *创建失败,提示信息创建失败,模态窗口不关闭,市场活动列表也不刷新

创建市场活动

点开创建按钮，跳出来模态窗口。

下拉列表里的是动态的，不能写死在页面里。都是从数据库里查出来的



模态窗口

模拟的动态窗口

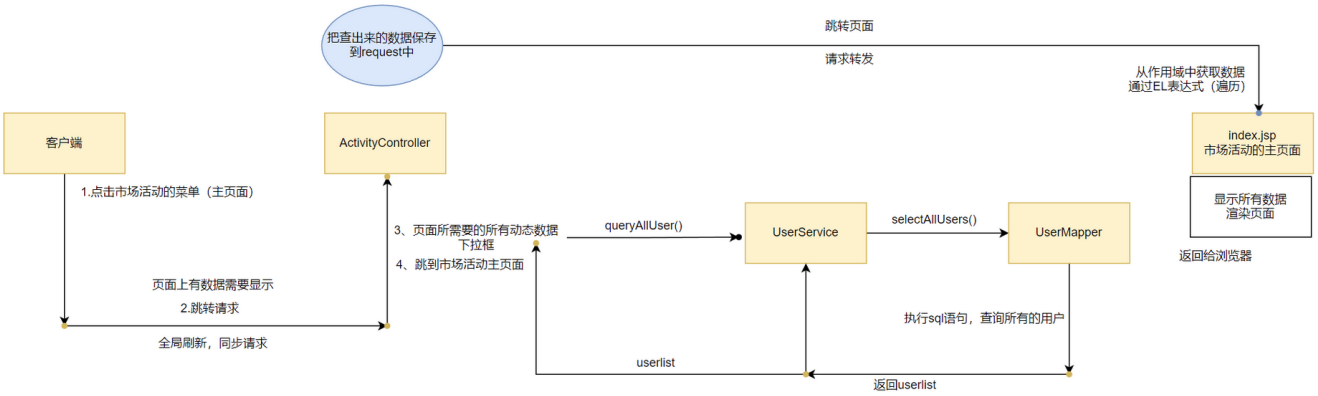
本质上是 ，不是显示一个独立的网页，在 里面显示一个独立的表单。通过设置z-index大小来实现的

初始时，z-index初始参数是<0，所以不显示。需要显示时，z-index值设置成>0即可。（前台框架控制）

模态窗口显示和隐藏

- **方式一：通过data-toggle="modal" data-target="模态窗口的id"**
- **方式二：通过js函数控制：**
  - **选择器（选中div）. modal("show"); 显示选中的模态窗**
  - **选择器（选中div）. modal("hide"); 关闭选中的模态窗口**
- **方式三：data-dismiss**

创建市场活动的流程



- **先写mapper：查询所有的用户**
  - ```
    /**
     * 查询所有的用户
     * @return
     */
    List<User> selectAllUsers();
    ```
- **xml里写sql语句：所有的用户，所有字段，从用户里查，提交到后台是把id提交到字段。至少需要两个字段：id、用户名、过滤掉禁用的：lock_state**
  - ```
    <select id="selectAllUsers" resultMap="BaseResultMap">
        select
        <include refid="Base_Column_List"/>
        from tbl_user
        where lock_state='1'
    </select>
    ```
- **下面写service层**
- ```
  public interface UserService {
      User queryUserByLoginActAndPwd(Map<String,Object> map);
      List<User> queryAllUsers();
  }
  ```
  ```

- **实现类impl：调mapper层**

```
@Service("userService")
public class UserServiceImpl implements UserService {

    @Autowired
    private UserMapper userMapper;

    @Override
    public User queryUserByLoginActAndPwd(Map<String, Object> map) {
        return userMapper.selectUserByLoginActAndPwd(map);
    }

    @Override
    public List<User> queryAllUsers() {
        return userMapper.selectAllUsers();
    }
}
```

- **写controller层**

```
@Controller
public class ActivityController {

    /**
     * 跳转到市场活动主页面
     */
    @Autowired
    private  UserService userService;

@RequestMapping("/workbench/activity/index.do")
    public String index(HttpServletRequest request){
        //调用service层方法查询所有的用户
        List<User>
```

```
userList=userService.queryAllUsers();
                                                    //把数据保存到request的
作用域中
                                                    //拿request对象，加参数
HttpServletRequest request

request.setAttribute("userList",userList);
                                                    //跳转到市场活动的主页
面：请求转发到市场活动的主页面
                                                    return
"/workbench/activity/index";
                                                }

                                            }
```

- **再写前台index代码**
  - **把用户遍历出来显示到下拉列表里**
    - ```
      <select class="form-control"
      id="create-marketActivityOwner">

      <!--遍历list,从集合里每拿一个元素放到这个u变量里去-->

      <c:forEach items="${userList}" var="u">

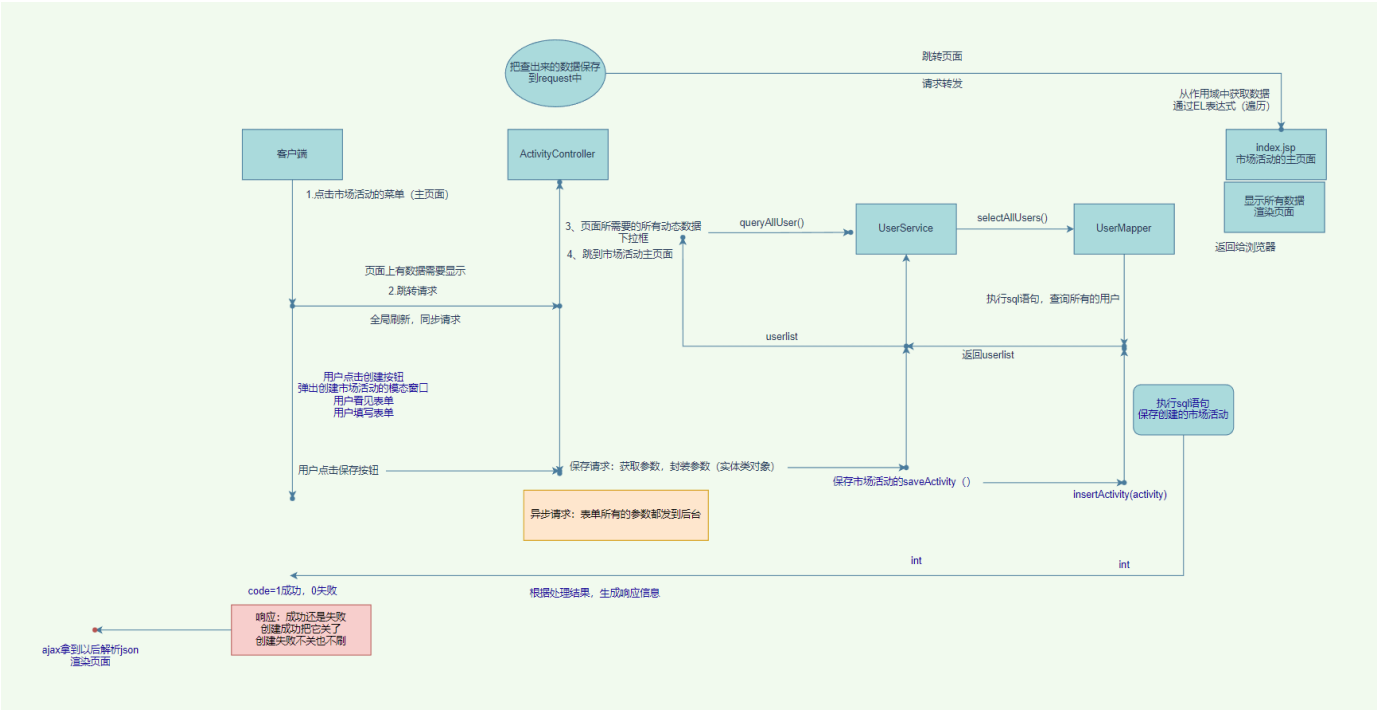
      <!--给用户看的是username,真正提交后台的是id-->

      <option value="${u.id}">${u.name}</option>

      </c:forEach>

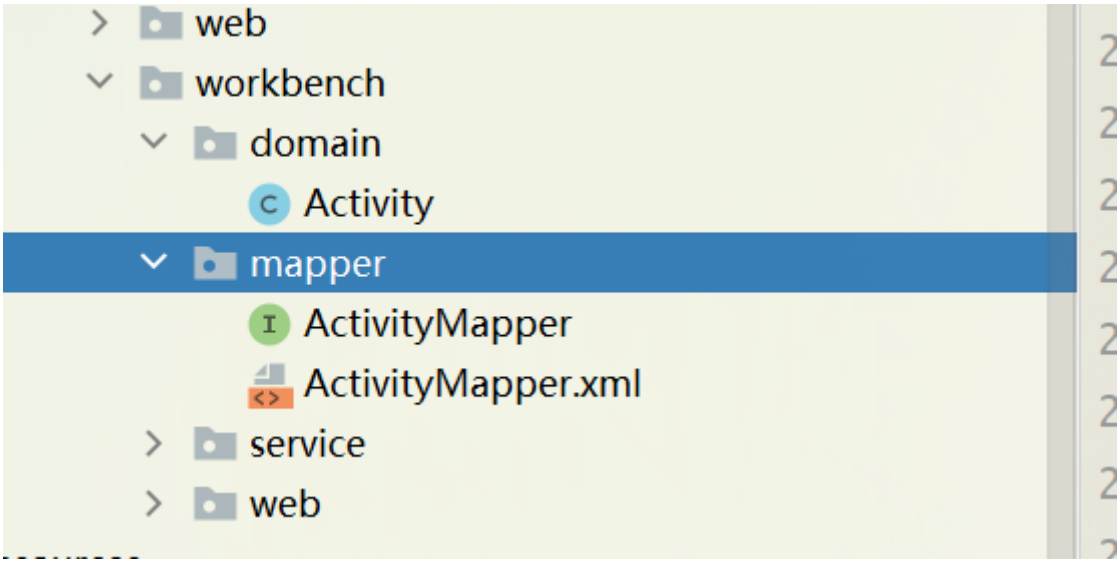
      </select>
      ```
    - 

      ```

- 改超级链接，变成controller


## 用户点击保存按钮

- 先写mapper：逆向工程，有表了生成代码

- 


- ```
  public interface ActivityMapper {

      int deleteByPrimaryKey(String id);

      int insertSelective(Activity record);

      Activity selectByPrimaryKey(String id);

      int updateByPrimaryKeySelective(Activity record);

      int updateByPrimaryKey(Activity record);

      /**
       * 保存创建的市场活动
       */
      int insertActivity(Activity activity);

  }
  ```

- **扫描**

- ```
  <!-- mapper注解扫描器配置,扫描@MapperScan注解,自动生成代码对象 -->
      <bean id="mapperScanner" class="org.mybatis.spring.mapper.MapperScannerConfigurer">
  ```

```
                <property name="basePackage"
value="com.bjpowernode.crm.settings.mapper,

com.bjpowernode.crm.workbench.mapper"/>
                <property
name="sqlSessionFactoryBeanName" value="sqlSessionFactory"/>
            </bean>
```

- **sql语句**

```
        <insert id="insertActivity"
parameterType="com.bjpowernode.crm.workbench.domain.Activity" >
            insert into tbl_activity (id,
owner, name, start_date,
            end_date, cost, description,
            create_time, create_by)
            values (#{id,jdbcType=CHAR}, #
{owner,jdbcType=CHAR}, #{name,jdbcType=VARCHAR}, #
{startDate,jdbcType=CHAR},
            #{endDate,jdbcType=CHAR}, #
{cost,jdbcType=VARCHAR}, #{description,jdbcType=VARCHAR},
            #{createTime,jdbcType=CHAR}, #
{createBy,jdbcType=VARCHAR})
        </insert>
```

- **service层：参数是实体类对象**

```
public interface ActivityService {
    int saveCreateActivity(Activity
activity);

}
```

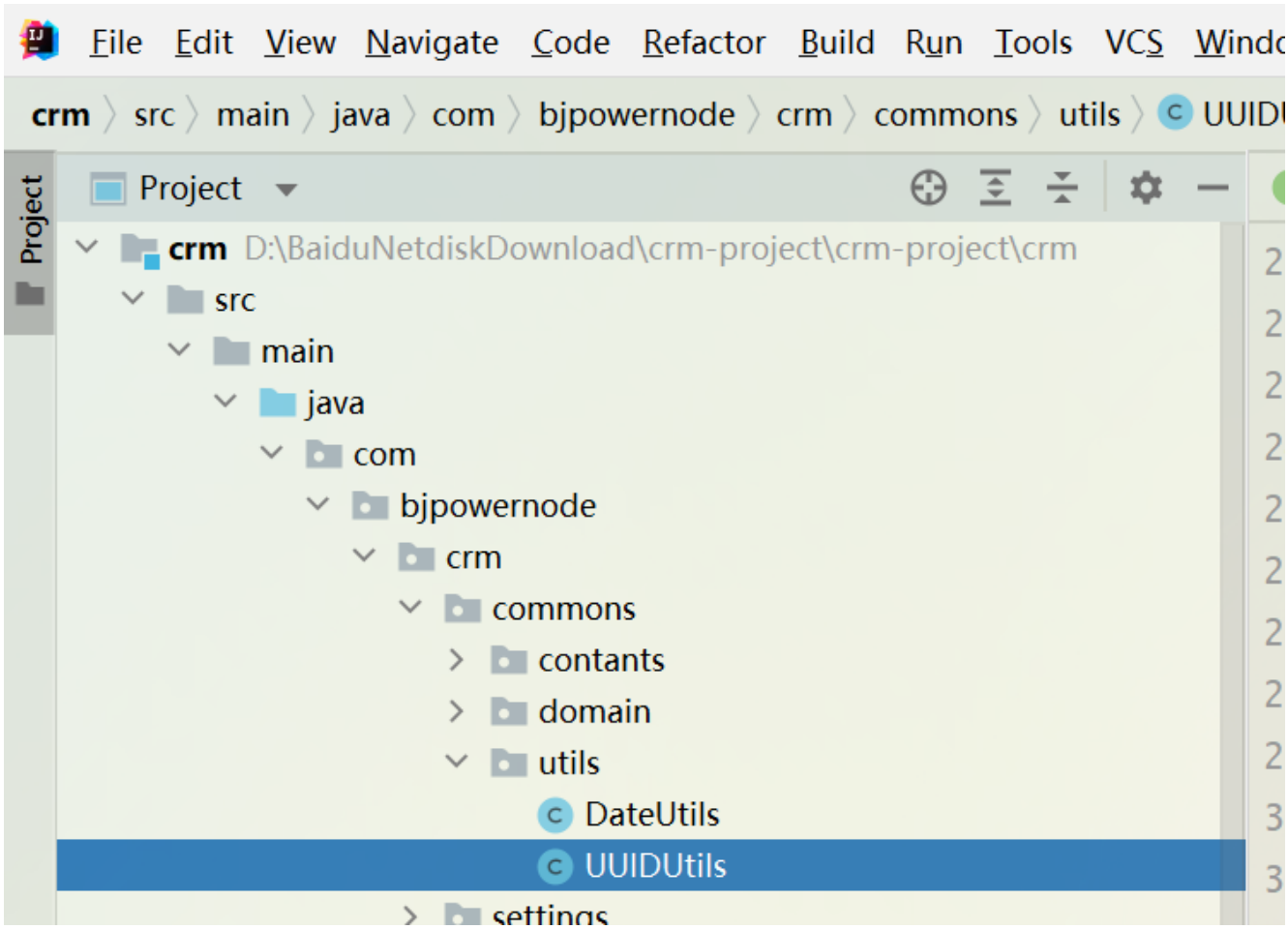
- **impl层**

```
@Service("activityService")
public class ActivityServiceImpl
implements ActivityService {


    @Autowired
    private ActivityMapper
activityMapper;


    @Override
```

```
    public int saveCreateActivity(Activity activity) {
        return activityMapper.insertActivity(activity);
    }
}
```

- **让spring扫描**

```
<context:component-scan base-package="com.bjpowernode.crm.workbench.service" />
```

- 

生成uuid

```
public class UUIDUtils {
    /**
     * 回去uuid的值
     * @return
     */
    public static String getUUID(){
        return UUID.randomUUID().toString().replaceAll("-","");
```

```
        }
    }
```

- **controller层**

```
    /**
     * 返回json字符串，封装成java对象。定义成父类，返回的子类都行。多态应用.
     * 响应信息返回到页面，一点击保存回到市场活动主页面。url和市场活动主页面保持一致
     */

@RequestMapping("/workbench/activity/saveCreateActivity.do")
    //要接受参数：直接定义成实体类形参，前台传过来的参数封装成实体类对象
    public @ResponseBody Object saveCreateActivity(Activity activity, HttpSession session) {
        //实体类只是封装了业务参数，进行二次封装
        User user = (User) session.getAttribute(Contants.SESSION_USER);//拿到当前用户

activity.setId(UUIDUtils.getUUID());

activity.setCreateTime(DateUtils.formateDateTime(new Date()));//当前的系统时间

activity.setCreateBy(user.getId());//谁创建的，谁点的保存按钮，谁登陆的，当前用户，session里的用户
        //考虑service层有没有报异常
        ReturnObject returnObject = new ReturnObject();
        try {
            //调用service层的方法，保存创建的市场活动，记录条数
            int ret = activityService.saveCreateActivity(activity);
            if (ret > 0) {

returnObject.setCode(Contants.RETURN_OBJECT_CODE_SUCCESS);//成功
            } else {

returnObject.setCode(Contants.RETURN_OBJECT_CODE_FAIL);//失败
```

```
returnObject.setMessage("系统繁忙，请稍后重试....");//失败信息
                }
            } catch (Exception e) {
                e.printStackTrace();
                returnObject.setMessage("系统繁忙，请稍后重试....");//失败信息
                //根据保存的结果生成响应信息
            }
            return returnObject;
        }
    }
```

前台页面

- **发请求**
  - **弹模态窗口：用id选择器，调modal函数，传个参数“show”**
    - ```
      <script type="text/javascript">

          $(function(){
              //给"创建"按钮添加单击事件
              $("#createActivityBtn").click(function () {
                  //初始化工作
                  //重置表单
                  $("#createActivityForm").get(0).reset();
                  //弹出创建市场活动的模态窗口
                  $("#createActivityModal").modal("show");
              });
      ```
  - **给保存按钮加单击事件:保存按钮去掉 data-dismiss，就不会关了。单击事件saveCreateActivityBtn**
    - ```
                          <div class="modal-footer">
                              <button type="button" class="btn btn-default" data-dismiss="modal">关闭</button>
                              <button type="button" class="btn btn-primary" id="saveCreateActivityBtn">保存</button>
                          </div>
      ```
    - ```
                          //给"保存"按钮添加单击事件

      $("#saveCreateActivityBtn").click(function () {
      ```
      ```

```
        });
    });
```

- **收集参数：表单里的参数值，拿到表单里value里的值**
  - ```
    //收集参数
    var owner=$("#create-marketActivityOwner").val();
    var name=$.trim($("#create-marketActivityName").val());
    var startDate=$("#create-startDate").val();
    var endDate=$("#create-endDate").val();
    var cost=$.trim($("#create-cost").val());
    var description=$.trim($("#create-description").val());
    ```

- **表单验证**
  - ```
    //表单验证
    if(owner==""){
        alert("所有者不能为空");
        return;
    }
    if(name==""){
        alert("名称不能为空");
        return;
    }
    if(startDate!=""&&endDate!=""){
        //使用字符串的大小代替日期的大小
        if(endDate<startDate){
            alert("结束日期不能比开始日期小");
            return;
        }
    }
    var regExp=/^(([1-9]\d*)|0)$/; //定义非负整数
    if(!regExp.test(cost)){ //匹配具体的字符串
    ```

```
                alert("成本只能为非负整数");
                return;
            }
```

- **发送请求：**

```
            //发送请求
            $.ajax({

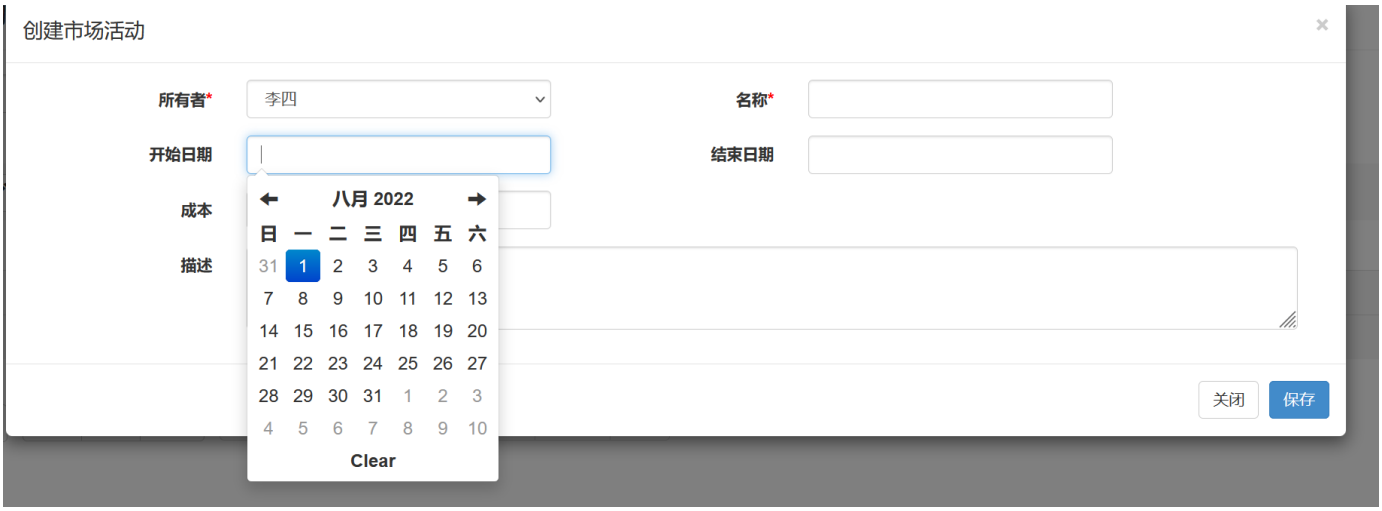
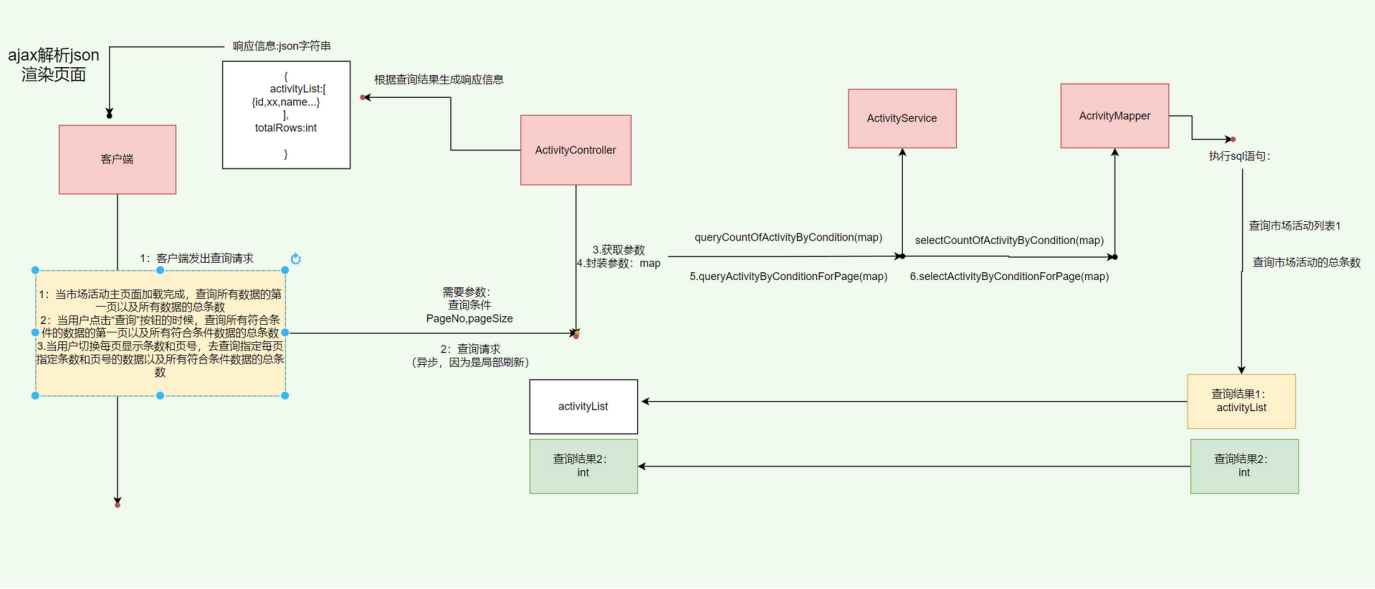
url:'workbench/activity/saveCreateActivity.do',    //请求发到controller里去
                data:{ //参数提交
                    owner:owner, //所有者，创建者

                    name:name, //市场活动名称
                    startDate:startDate, //开始日期

                    endDate:endDate, //结束日期
                    cost:cost,  //成本
                    description:description //描述

                },
                type:'post',
                dataType:'json',
```

- **处理响应**

```
                //处理响应
                success:function (data) {
                    if(data.code=="1"){ //成功了
                        //关闭模态窗口

$("#createActivityModal").modal("hide");
                        //刷新市场活动列，显示第一页数据，保持每页显示条数不变(保留)
                    }else{
                        //提示信息
                        alert(data.message);
                        //模态窗口不关闭

$("#createActivityModal").modal("show");//可以不写。
                    }
                }
```

添加日历功能

```
<div class="form-group">
                            <label for="create-startDate" class="col-sm-2 control-label">开始日期</label>
                            <div class="col-sm-10" style="width: 300px;">
                                <input type="text" class="form-control" id="create-startDate">
                            </div>
                            <label for="create-endDate" class="col-sm-2 control-label">结束日期</label>
                            <div class="col-sm-10" style="width: 300px;">
                                <input type="text" class="form-control" id="create-endDate">
                            </div>
                        </div>
```

给容器加上单击事件调用工具函数

```
//当容器加载完成之后，对容器调用工具函数
        $(".mydate").datetimepicker({
            language:"zh-CN",//语言
            format:'yyyy-mm-dd',//日期的格式
            minView:'month',//可以选择的最小视图
            initialDate:new Date(),//初始化显示的日期
            autoclose:true,//设置选择完日期或者时间之后，是否自动关闭日历
            clearBtn:true,//设置是否显示清空按钮
        });
```



分页查询市场活动

查询市场活动

当市场活动主页面加载完成之后,显示所有数据的第一页;

用户在市场活动主页面填写查询条件,点击"查询"按钮,显示所有符合条件的数据的第一页，保持每页显示条数不变

实现翻页功能.

- 在市场活动主页面,显示市场活动列表和记录的总条数
- 默认每页显示条数:10



- 先写mapper层

```
    /**
     * 根据条件分页查询
     * @param map
     * @return
     */
    List<Activity> selectActivityByConditionForPage(Map<String,Object> map);
```

```
    <select id="selectActivityByConditionForPage" parameterType="map" resultMap="BaseResultMap">
        select
        a.id,
        u1.name as owner,
        a.name,
        a.start_date,
        a.end_data,
        a.cost,a.description,a.create_time,
```

```
                u2.name as create_by
                a.edit_time,
                u3.name as edit_by
                from tbl_activity a
                <!--连接用户表：内连接-->
                join tb_user u1 on a.owner=u1.id
                join tb_user u2 on a.create_by=u2.id
                left join tbl_user u3 on a.edit_by=u3.id
                <!--动态sql-->
                <where>
                  <if test="name!=null and name!=''">
                    and a.name like '%' #{name} '%'
                  </if>

                  <if test="owner!=null and owner!=''">
                    and u1.name like '%' #{owner} '%'
                  </if>

                  <if test="owner!=null and owner!=''">
                    and u1.name like '%' #{owner} '%'
                  </if>

                  <if test="start_date!=null and start_date!=''">
                    and  a.start_date>=#{startDate}
                  </if>

                  <if test="end_data!=null and end_data!=''">
                    and  a.end_data&lt;=#{endDate}
                  </if>

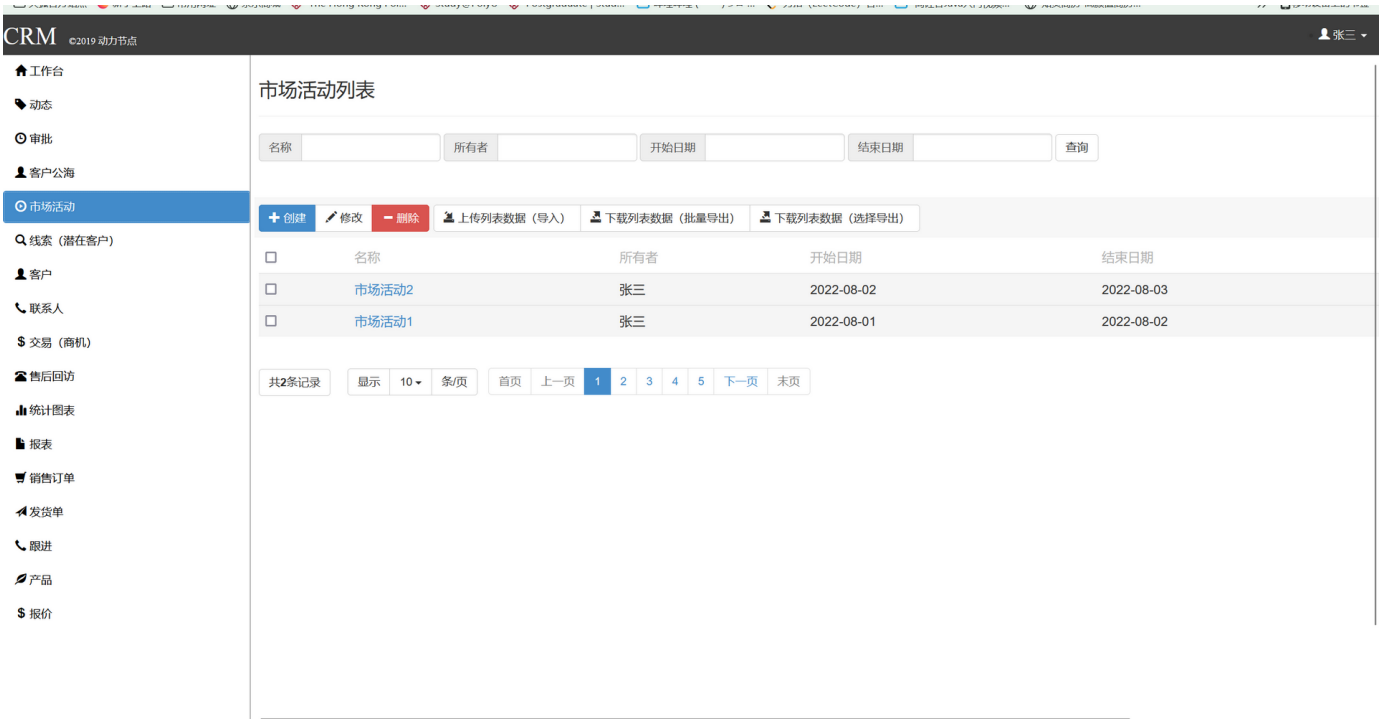
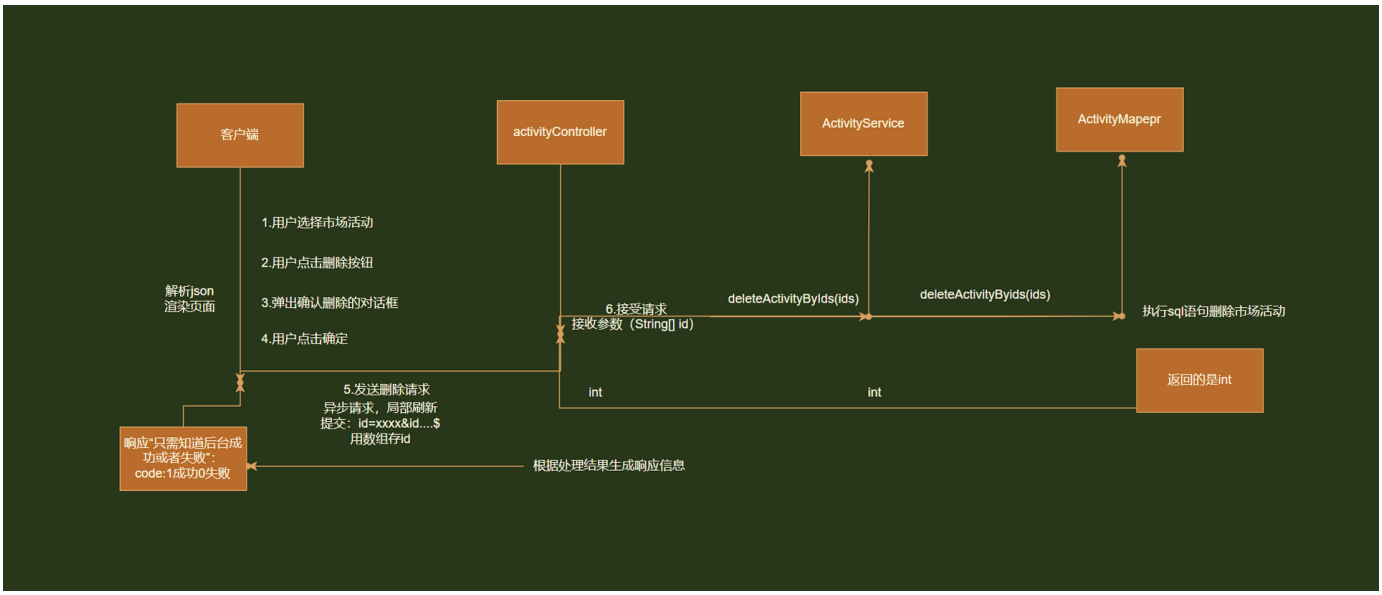
                </where>
                    order by a.create_time desc
                    <!--分页-->
                    limit #{beginNo},#{pageSize}

              </select>
```

- 写service层
  - ```
    List<Activity> queryActivityByConditionForPage(Map<String,Object> map);
    ```
  - serviceImple
    ```

```
@Override
    public List<Activity> queryActivityByConditionForPage(Map<String, Object> map) {

        return activityMapper.selectActivityByConditionForPage(map);
    }
```

- mapper层查询总条数
  - ```
    int selectCountOfActivityByCondition(Map<String,Object> map);
    ```
  - sql语句
    - ```
      <select id="selectActivityByConditionForPage" parameterType="map" resultMap="BaseResultMap">
        select count(*)
        from tbl_activity a
        <!--连接用户表：内连接-->
        join tb_user u1 on a.owner=u1.id
        join tb_user u2 on a.create_by=u2.id
        left join tbl_user u3 on a.edit_by=u3.id
        <!--动态sql-->
        <where>
          <if test="name!=null and name!=''">
            and a.name like '%' #{name} '%'
          </if>

          <if test="owner!=null and owner!=''">
            and u1.name like '%' #{owner} '%'
          </if>

          <if test="owner!=null and owner!=''">
            and u1.name like '%' #{owner} '%'
          </if>

          <if test="start_date!=null and start_date!=''">
            and  a.start_date>=#{startDate}
          </if>

          <if test="end_data!=null and end_data!=''">
            and  a.end_data<=#{endDate}
      ```

```
            </if>
        </where>
    </select>
```

- service层
  - ```
    int queryCountOfActivityByCondition(Map<String,Object> map);
    ```
  - 实现
    - ```
      @Override
      public int queryCountOfActivityByCondition(Map<String, Object> map) {
          return activityMapper.selectCountOfActivityByCondition(map);
      }
      ```
- controller层
  - ```
    @RequestMapping("/workbench/activity/queryActivityByConditionForPage.do")
    public @ResponseBody Object queryActivityByConditionForPage(String name,String owner,String startDate,String endDate,

    int pageNo,int pageSize){
        //封装参数
        Map<String,Object> map=new HashMap<>();
        map.put("name",name);
        map.put("owner",owner);
        map.put("startDate",startDate);
        map.put("endDate",endDate);
        map.put("beginNo",(pageNo-1)*pageSize);
        map.put("pageSize",pageSize);
        //调用service层方法，查询数据
        List<Activity> activityList=activityService.queryActivityByConditionForPage(map);
        int totalRows=activityService.queryCountOfActivityByCondition(map);
        //根据查询结果结果，生成响应信息
        Map<String,Object> retMap=new HashMap<>();
        retMap.put("activityList",activityList);
    ```
    ```

```
            retMap.put("totalRows",totalRows);
            return retMap;
        }
```



翻页功能

```
function queryActivityByConditionForPage(pageNo,pageSize) {
    //收集参数
    var name=$("#query-name").val();
    var owner=$("#query-owner").val();
    var startDate=$("#query-startDate").val();
    var endDate=$("#query-endDate").val();
//  var pageNo=1;不能写死
//  var pageSize=10;
    //发送请求
```

封装

实现翻页功能.

*在市场活动主页面,显示市场活动列表和记录的总条数

*默认每页显示条数:10

下拉列表加一个change事件

分页插件：bs_pagination插件

- 当页面刚加载完
  - queryActivityByConditionForPage（1,10）：
    把pageNo，pageSize和查询条件一起发送到后台，查询数据
    data：返回
    activityIIST:遍历，list显示列表
    totalRows：调用工具函数，显示翻页信息
- 当用户切换页号或者每页显示条数时
  - pageNo,pageSize变了：翻页信息会自动变化
  - 手动刷新列表：把pageNo，pageSize和查询条件一起发到后台，查询数据
  - 又返回data：
    - activityList：遍历list，显示列表
    - totalRows：调用工具函数，显示翻页信

删除市场活动

用户在市场活动主页面,选择要修改的市场活动,点击"修改"按钮,弹出修改市场活动的模态窗口;

用户在修改市场活动的模态窗口填写表单,点击"更新"按钮,完成修改市场活动的功能.

*每次能且只能修改一条市场活动

*所有者 动态的

*表单验证(同创建)

*修改成功之后,关闭模态窗口,刷新市场活动列表,保持页号和每页显示条数都不变

*修改失败,提示信息,模态窗口不关闭,列表也不刷新



全选和取消

给全选按钮添加单击事件

```
//给"全选"按钮添加单击事件
        $("#chckAll").click(function () {
            //如果"全选"按钮是选中状态，则列表中所有checkbox都选中
            //this：代表当前正在发生事件的对象
            $("#tBody input[type='checkbox']").prop("checked",this.checked);
        });
        $("#tBody").on("click","input[type='checkbox']",function () {
            //如果列表中的所有checkbox都选中，则"全选"按钮也选中
            if($("#tBody input[type='checkbox']").size()==$("#tBody
input[type='checkbox']:checked").size()){
                $("#chckAll").prop("checked",true);
            }else{//如果列表中的所有checkbox至少有一个没选中，则"全选"按钮也取消
                $("#chckAll").prop("checked",false);
            }
        });
```

删除市场活动的mapper层

```
/**
     * 根据ids批量删除市场活动
     *
     * @param ids
     * @return
     */
    int deleteActivityByIds(String[] ids);
```<delete id="deleteActivityByIds" parametertype="string">delete from tbl_activity where id in <foreach collection="array" item="id" separator="," open="(" close=")">#{id}</foreach></delete> ```

自动返回影响记录条数，不用再加resultmap

根据id删，多条不能写=，写<foreach>遍历集合数组会拼成一个字符串。

- ## service层

```java
int deleteActivityByIds(String[] ids);
```

- **impl层**

```
        @Override
            public int deleteActivityByIds(String[] ids) {
                return activityMapper.deleteActivityByIds(ids);
            }
```

- controller层

```
        @RequestMapping("/workbench/activity/deleteActivityIds.do")
            public @ResponseBody Object deleteActivityIds(String[] id){
                //形参String[] id: 接受前台发来的数组
                ReturnObject returnObject=new ReturnObject();//
                try {
                    //调用service层方法，删除市场活动，ret：影响记录条数
                    int ret = activityService.deleteActivityByIds(id);

                    if(ret>0){//如果影响记录条数>0

returnObject.setCode(Contants.RETURN_OBJECT_CODE_SUCCESS);//删除成功
                    }else{

returnObject.setCode(Contants.RETURN_OBJECT_CODE_FAIL);//删除失败
                        returnObject.setMessage("系统忙，请稍后重试....");
                    }
                }catch (Exception e){
                    e.printStackTrace();
                    returnObject.setCode(Contants.RETURN_OBJECT_CODE_FAIL);
                    returnObject.setMessage("系统忙，请稍后重试....");
                }
                return returnObject;
            }
```

- 前端页面

```
        //给"删除"按钮添加单击事件
                $("#deleteActivityBtn").click(function () {
                    //收集参数
                    //获取列表中所有被选中的checkbox
                    var chekkedIds=$("#tBody
input[type='checkbox']:checked");
                    if(chekkedIds.size()==0){
                        alert("请选择要删除的市场活动");
```

```
                return;
            }

            if(window.confirm("确定删除吗？")){
                var ids="";
                $.each(chekkedIds,function ()
{//id=xxxx&id=xxx&.....&id=xxx&
                    ids+="id="+this.value+"&";
                });
                ids=ids.substr(0,ids.length-
1);//id=xxxx&id=xxx&.....&id=xxx

                //发送请求
                $.ajax({
                    url:'workbench/activity/deleteActivityIds.do',
                    data:ids,
                    type:'post',
                    dataType:'json',
                    success:function (data) {
                        if(data.code=="1"){
                            //刷新市场活动列表,显示第一页数据,保持
每页显示条数不变

queryActivityByConditionForPage(1,$("#demo_pag1").bs_pagination('getOption',
'rowsPerPage'));
                        }else{
                            //提示信息
                            alert(data.message);
                        }
                    }
                });
            }
        });
```

修改

用户在市场活动主页面,选择要修改的市场活动,点击"修改"按钮,弹出修改市场活动的模态窗口;

用户在修改市场活动的模态窗口填写表单,点击"更新"按钮,完成修改市场活动的功能.

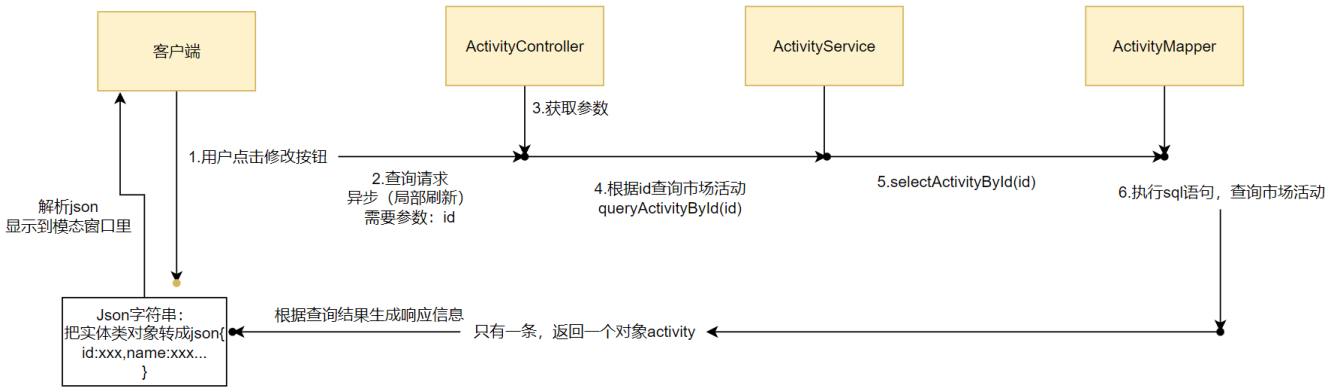
***每次能且只能修改一条市场活动**

***所有者 动态的**

***表单验证(同创建)**

***修改成功之后,关闭模态窗口,刷新市场活动列表,保持页号和每页显示条数都不变**

***修改失败,提示信息,模态窗口不关闭,列表也不刷新**



mapper层:

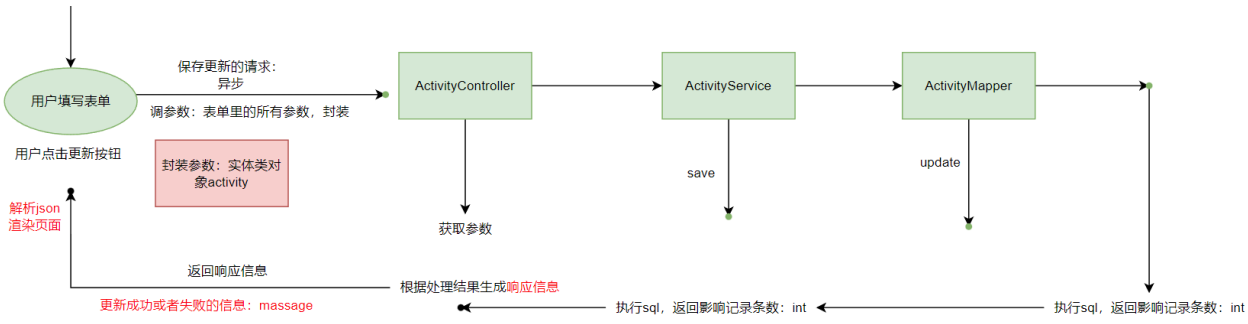
- ```
  /**
   * 根据id查询市场活动的信息
   *
   * @param id
   * @return
   */
  Activity selectActivityById(String id);
  ```

sql语句

- ```
  <select id="selectActivityById" parameterType="string" resultMap="BaseResultMap">
    select
    <include refid="Base_Column_List"/>
    from tbl_activity
    where id=#{id}
  </select>
  ```

Service层

- ```
  Activity queryActivityById(String id);
  ```
- impl层
  ```

```
    @Override
        public Activity queryActivityById(String id) {
            return activityMapper.selectActivityById(id);
        }
```

controller层

```
    @RequestMapping("/workbench/activity/queryActivityById.do")
        public @ResponseBody Object queryActivityById(String id){
            //调用service层方法，查询市场活动
            Activity activity=activityService.queryActivityById(id);
            //根据查询结果，返回响应信息
            return activity;
        }
```

前端

```
    //给"修改"按钮添加单击事件
            $("#editActivityBtn").click(function () {
                //收集参数:id
                //获取列表中被选中的checkbox

                var chkedIds=$("#tBody input[type='checkbox']:checked");//被选中的checkbox
                if(chkedIds.size()==0){
                    alert("请选择要修改的市场活动");
                    return;
                }
                if(chkedIds.size()>1){
                    alert("每次只能修改一条市场活动");
                    return;
                }
                //var id=chkedIds.val();
                //var id=chkedIds.get(0).value;
                var id=chkedIds[0].value; //获取value值
                //发送请求
                $.ajax({
                    url:'workbench/activity/queryActivityById.do',
                    data:{
                        id:id
                    },
                    type:'post',
```

```
                dataType:'json',
                success:function (data) {
                    //把市场活动的信息显示在修改的模态窗口上
                    $("#edit-id").val(data.id);
                    $("#edit-marketActivityOwner").val(data.owner);
                    $("#edit-marketActivityName").val(data.name);
                    $("#edit-startTime").val(data.startDate);
                    $("#edit-endTime").val(data.endDate);
                    $("#edit-cost").val(data.cost);
                    $("#edit-description").val(data.description);
                    //弹出模态窗口
                    $("#editActivityModal").modal("show");
                }
            });
        });
```

封装参数

- 如果做查询条件，或者参数之间不属于一个实体类对象，封装成map
- 如果做写数据，并且参数本来就是属于一个实体类对象的，封装成实体类对象

更新表单



mapper层

- ```
    /**
     * 保存修改的市场活动
     *
     * @param activity
     * @return
     */
    int updateActivity(Activity activity);
  ```

sql语句
  ```

- ```
  <update id="updateActivity" parameterType="com.bjpowernode.crm.workbench.domain.Activity">
          update tbl_activity
          set owner=#{owner},name=#{name},start_date=#{startDate},end_date=#{endDate},cost=#{cost},description=#{description},
              edit_time=#{editTime},edit_by=#{editBy}
          where id=#{id}
      </update>
  ```

service层

- ```
  int saveEditActivity(Activity activity);
  ```
- impl层
  - ```
    @Override
        public int saveEditActivity(Activity activity) {
            return activityMapper.updateActivity(activity);
        }
    ```

controller层

- ```
  @RequestMapping("/workbench/activity/saveEditActivity.do")
      public @ResponseBody Object saveEditActivity(Activity activity,HttpSession session){
          //获得当前的user
          User user=(User) session.getAttribute(Contants.SESSION_USER);
          //封装参数
          activity.setEditTime(DateUtils.formateDateTime(new Date()));
          activity.setEditBy(user.getId());

          ReturnObject returnObject=new ReturnObject();
          try {
              //调用service层方法，保存修改的市场活动:ret影响记录条数
              int ret = activityService.saveEditActivity(activity);
              //根据处理结果生成响应信息：写成功了还是写失败了
              if(ret>0){
                  returnObject.setCode(Contants.RETURN_OBJECT_CODE_SUCCESS);
              }else{
                  returnObject.setCode(Contants.RETURN_OBJECT_CODE_FAIL);
                  returnObject.setMessage("系统忙，请稍后重试....");
              }
          }catch (Exception e){
  ```

```
            e.printStackTrace();
            returnObject.setCode(Contants.RETURN_OBJECT_CODE_FAIL);
            returnObject.setMessage("系统忙，请稍后重试....");
        }
        return returnObject;
    }
```

前端

```
//给"更新"按钮添加单击事件
        $("#saveEditActivityBtn").click(function () {
            //收集参数
            var id=$("#edit-id").val();
            var owner=$("#edit-marketActivityOwner").val();
            var name=$.trim($("#edit-marketActivityName").val());
            var startDate=$("#edit-startTime").val();
            var endDate=$("#edit-endTime").val();
            var cost=$.trim($("#edit-cost").val());
            var description=$.trim($("#edit-description").val());
            //表单验证(作业)
            //发送请求
            $.ajax({
                url:'workbench/activity/saveEditActivity.do',
                data:{
                    id:id,
                    owner:owner,
                    name:name,
                    startDate:startDate,
                    endDate:endDate,
                    cost:cost,
                    description:description
                },
                type:'post',
                dataType:'json',
                success:function (data) {
                    if(data.code=="1"){
                        //关闭模态窗口
                        $("#editActivityModal").modal("hide");
                        //刷新市场活动列表,保持页号和每页显示条数都不变

queryActivityByConditionForPage($("#demo_pag1").bs_pagination('getOption',
'currentPage'),$("#demo_pag1").bs_pagination('getOption', 'rowsPerPage'));
                    }else{
```

```
                    //提示信息
                    alert(data.message);
                    //模态窗口不关闭
                    $("#editActivityModal").modal("show");
                }
            }
        });
    });
```