营销

> *做为售卖课程为主要收入的应用，需要有营销功能，让用户帮忙推广，给予用户一定的佣金，这样网站才能长久发展，用户规模才能慢慢扩大*

1. 生成推广链接

> *每个用户有一个唯一的推广链接，链接当中含有此用户的标识，当被推广用户点击链接时，就会和推广人产生关联*

1.1 API

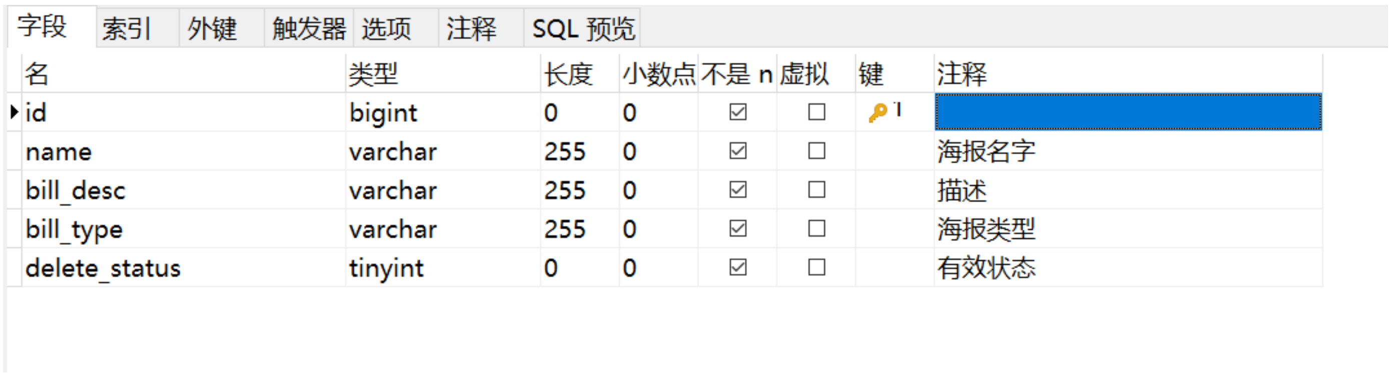
请求url：/api/i/gen

请求参数：{"id":1}

返回数据:

```
{
    "code":2000000000,
    "message":"default success",
    "result":"http://www.mszlu/com/invite/i/u/1/加密字符串",
    "success":true
}
```

1.2 数据库设计



名	类型	长度	小数点	不是 null	虚拟	键	注释
id	bigint	0	0	☑	☐	🔑1	
name	varchar	255	0	☑	☐		海报名字
bill_desc	varchar	255	0	☑	☐		描述
bill_type	varchar	255	0	☑	☐		海报类型
delete_status	tinyint	0	0	☑	☐		有效状态

说明：

推广链接生成后，前端会生成一张海报图片，推广链接以二维码的形式展示在海报上

```java
package com.xiaopizhu.tiku.dao.data;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class Bill {

    private Long id;

    private String name;

    private String billDesc;

    private String billType;

    private Integer deleteStatus;

}
```

1.3 编码

1.3.1 Controller

```java
package com.mszlu.xt.web.api;

import com.mszlu.common.model.CallResult;
import com.mszlu.xt.model.service.BillService;
import com.mszlu.xt.web.model.params.BillParam;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;

@Controller
@RequestMapping("i")
public class InviteApi {

    @Autowired
    private BillService billService;
    @RequestMapping("gen")
    @ResponseBody
    public CallResult gen(@RequestBody BillParam billParam){
        return billService.gen(billParam);
    }
}
```

```java
package com.mszlu.xt.web.model.params;

import lombok.Data;

@Data
public class BillParam {

    private Integer page = 1;
    private Integer pageSize = 10;

    private Long id;
    private String name;
    private String billDesc;
    private String billType;
    /**
     * 0 正常 1 删除
     */
    private Integer status;

    private Long userId;
}
```

1.3.2 Service

```
package com.mszlu.xt.model.service;

import com.mszlu.common.model.CallResult;
import com.mszlu.xt.web.model.params.BillParam;

public interface BillService {
    CallResult gen(BillParam billParam);
}
```

```
package com.mszlu.xt.web.service.impl;

import com.mszlu.common.model.CallResult;
import com.mszlu.xt.model.service.BillService;
import com.mszlu.xt.web.domain.BillDomain;
import com.mszlu.xt.web.domain.repository.BillDomainRepository;
import com.mszlu.xt.web.model.params.BillParam;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

@Service
@Transactional
public class BillServiceImpl extends AbstractService implements
BillService {
    @Autowired
    private BillDomainRepository billDomainRepository;

    @Override
    public CallResult gen(BillParam billParam) {
        BillDomain billDomain =
this.billDomainRepository.createDomain(billParam);
        return billDomain.gen();
    }
}
```

1.3.3 Domain

用到的工具类:

```
<dependency>
      <groupId>com.xiaoleilu</groupId>
      <artifactId>hutool-all</artifactId>
      <version>3.0.9</version>
    </dependency>
```

```
package com.mszlu.common.utils;

import com.xiaoleilu.hutool.crypto.SecureUtil;
import com.xiaoleilu.hutool.crypto.symmetric.AES;

public class AESUtils {
    public static String key = "lzxttyuiopasdfgh";

    public static String encrypt(String string){

        AES aes = SecureUtil.aes(key.getBytes());
        //加密为16进制表示
        String encryptHex = aes.encryptHex(string);
        return encryptHex;
    }

    public static String decrypt(String string){

        AES aes = SecureUtil.aes(key.getBytes());
        //加密为16进制表示
        String decryptStr = aes.decryptStr(string);
        return decryptStr;
    }
    public static String cookieInviteId(String billType){
        return  AESUtils.encrypt("lzxt_invite_id_"+billType);
    }

    public static void main(String[] args) {
        Integer a = Integer.valueOf(200);
        Integer b = Integer.valueOf(200);
        System.out.println(a == b);
    }
}
```

```
package com.mszlu.xt.web.domain.repository;

import com.mszlu.xt.pojo.Bill;
import com.mszlu.xt.web.dao.BillMapper;
import com.mszlu.xt.web.domain.BillDomain;
import com.mszlu.xt.web.model.params.BillParam;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

import javax.annotation.Resource;

@Component
public class BillDomainRepository {


    @Value("invite.url")
    public String inviteUrl;

    public BillDomain createDomain(BillParam billParam){
        return new BillDomain(this,billParam);
    }
    @Resource
    private BillMapper billMapper;

    public Bill findBill(Long id) {
        return billMapper.selectById(id);
    }
}
```

```
package com.mszlu.xt.web.domain;

import com.mszlu.common.model.CallResult;
import com.mszlu.common.utils.AESUtils;
import com.mszlu.common.utils.UserThreadLocal;
import com.mszlu.xt.pojo.Bill;
import com.mszlu.xt.web.domain.repository.BillDomainRepository;
```

```java
import com.mszlu.xt.web.model.params.BillParam;

public class BillDomain {
    private BillDomainRepository billDomainRepository;
    private BillParam billParam;

    public BillDomain(BillDomainRepository billDomainRepository,
BillParam billParam) {
        this.billDomainRepository = billDomainRepository;
        this.billParam = billParam;
    }

    public CallResult gen() {
        Long id = this.billParam.getId();
        //根据传递的参数 获取海报信息
        Bill bill = this.billDomainRepository.findBill(id);
        if (bill == null){
            return CallResult.fail(-999,"id 不存在");
        }
        Long userId = UserThreadLocal.get();
        //将用户id做加密处理，AES对称加密算法，可以解密
        String userIdStr = AESUtils.encrypt(String.valueOf(userId));

        return CallResult.success(this.billDomainRepository.inviteUrl
+bill.getBillType()+"/"+userIdStr);
    }
}
```

```properties
invite.url=http://www.mszlu.com/api/i/u/
```

1.4 测试

2. 访问推广链接

> *在用户登录的情况下，访问推广链接，将登录用户和推广链接中携带的推广人id进行关联，如果被推广人产生登录，购买行为，记录推广的信息*

2.1API

请求url：u/{billType}/{id}

请求参数：路径参数

返回数据: 跳转页面

2.2 逻辑

1. 根据路径中的参数，获取海报类型和推荐人id
2. 将其加密后，存入cookie中
3. 当被推荐人登录或者购买时，就可以从cookie中获取推荐人的信息
4. 跳转页面

2.3 编码

```
@RequestMapping("u/{billType}/{id}")
    public String url(HttpServletRequest request, HttpServletResponse
response, @PathVariable("billType") String billType, @PathVariable("id")
String id){
        if (id != null){
            try {
                //key需要设置一个用户无法识别的，这样更好的隐藏意图
                Cookie cookies = new Cookie("_i_ga_b_"+billType,id);
                //设置时长为3天
                cookies.setMaxAge(3*24*60*60);
                cookies.setPath("/");
                response.addCookie(cookies);
            }catch (Exception e){
                e.printStackTrace();
            }
        }
        String ua = request.getHeader("user-agent").toLowerCase();
        if (ua.indexOf("micromessenger") > 0){
            //微信浏览器 跳转微信登录
            return "redirect:/wechat/authorize";
        }
        return "redirect:http://www.mszlu.com";
    }
```

3. 获取所有的海报

> *获取所有可用的海报信息，便于用户根据海标生成对应带有二维码的海报*

3.1 API

请求url: /api/i/all

请求参数:无 get

返回数据:

```java
package com.mszlu.xt.web.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class BillModel {

    private Long id;

    private String name;

    private String billDesc;

    private String billType;

}
```

3.2 编码

3.2.1 Controller

```
@RequestMapping("all")
    @ResponseBody
    public CallResult  all(){
        return billService.all(new BillParam());
    }
```

3.2.2 Service

```
CallResult all(BillParam billParam);
```

```
@Override
    public CallResult all(BillParam billParam) {
        BillDomain billDomain =
this.billDomainRepository.createDomain(billParam);
        return this.serviceTemplate.executeQuery(new
AbstractTemplateAction<Object>() {
            @Override
            public CallResult<Object> doAction() {
                return billDomain.findAllBillModelList();
            }
        });
    }
```

3.3.3 Domain

```
public CallResult<Object> findAllBillModelList() {
        List<Bill> billList = this.billDomainRepository.findBillList();
        List<BillModel> billModelList = new ArrayList<>();
        BeanUtils.copyProperties(billList,billModelList);
        return CallResult.success(billModelList);
    }
```

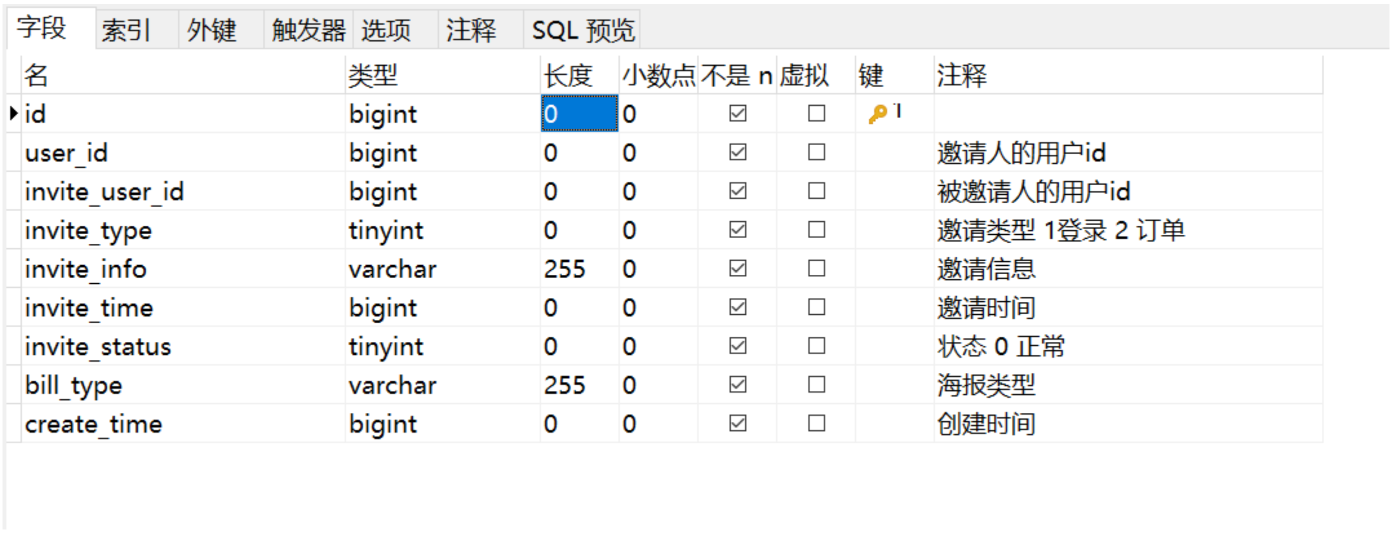
```
public List<Bill> findBillList() {

        return billMapper.selectList(Wrappers.lambdaQuery());
    }
```

3.3 测试

4. 修改登录逻辑

> *如果用户登录，获取cookie中的信息，如果cookie中有推荐人的信息，并且此用户是首次登录（新用户），那么标明此用户为推荐人推荐注册的*

4.1 数据库设计



名	类型	长度	小数点	不是 null	虚拟	键	注释
id	bigint	0	0	☑	☐	🔑1	
user_id	bigint	0	0	☑	☐		邀请人的用户id
invite_user_id	bigint	0	0	☑	☐		被邀请人的用户id
invite_type	tinyint	0	0	☑	☐		邀请类型 1登录 2 订单
invite_info	varchar	255	0	☑	☐		邀请信息
invite_time	bigint	0	0	☑	☐		邀请时间
invite_status	tinyint	0	0	☑	☐		状态 0 正常
bill_type	varchar	255	0	☑	☐		海报类型
create_time	bigint	0	0	☑	☐		创建时间

```java
package com.mszlu.xt.pojo;

import lombok.Data;

@Data
public class Invite {

    private Long id;
    private Long userId;
    private Long inviteUserId;
    private Integer inviteType;
    private String inviteInfo;
    private Long inviteTime;
    private Integer inviteStatus;
    private String billType;
    private Long createTime;
}
```

```java
package com.mszlu.xt.model.enums.invite;

import java.util.HashMap;
import java.util.Map;

public enum InviteType {
    /**
     * look name
     */
    LOGIN(1,"wx 登录"),
    ORDER(2,"order");

    private static final Map<Integer, InviteType> CODE_MAP = new
HashMap<>(3);

    static{
        for(InviteType topicType: values()){
            CODE_MAP.put(topicType.getCode(), topicType);
        }
    }

    /**
     * 根据code获取枚举值
     * @param code
     * @return
     */
    public static InviteType valueOfCode(int code){
        return CODE_MAP.get(code);
    }

    private int code;
    private String msg;

    InviteType(int code, String msg) {
        this.code = code;
        this.msg = msg;
    }

    public int getCode() {
        return code;
    }

    public void setCode(int code) {
        this.code = code;
    }

    public String getMsg() {
        return msg;
    }

    public void setMsg(String msg) {
        this.msg = msg;
    }
}
```

4.2 编码

```
  if (user == null){
                        user = saveUser(wxMpUser, registerTime);
                        //新用户 需要判断是否有邀请信息
                        fillInvite(user);
                    }
```

```
private void fillInvite(User user) {
        HttpServletRequest request = this.loginParam.getRequest();
        Cookie[] cookies = request.getCookies();
      if(cookies == null){
            return;
        }
        List<Map<String,String>> billTypeList = new ArrayList<>();
        for (Cookie cookie : cookies) {
            String name = cookie.getName();
            String[] inviteCookie = name.split("_i_ga_b_");
            if (inviteCookie.length == 2){
                Map<String,String> map = new HashMap<>();
                map.put("billType",inviteCookie[1]);
                map.put("userId",cookie.getValue());
                billTypeList.add(map);
            }
        }
        for (Map<String,String> inviteMap : billTypeList) {
            //有推荐信息，构建邀请信息
            Invite invite = new Invite();
            invite.setInviteInfo(user.getUnionId());
            invite.setInviteStatus(0);
            invite.setInviteTime(System.currentTimeMillis());
            invite.setInviteType(InviteType.LOGIN.getCode());
            invite.setInviteUserId(user.getId());

 invite.setUserId(Long.parseLong(AESUtils.decrypt(inviteMap.get("userId"
))));
            invite.setBillType(inviteMap.get("billType"));
            invite.setCreateTime(System.currentTimeMillis());

 this.loginDomainRepository.createInviteDomain(null).save(invite);
        }
    }
```

LoginDomainRepository:

```
@Autowired
    private InviteDomainRepository inviteDomainRepository;

    public InviteDomain createInviteDomain(InviteParam inviteParam) {

        return inviteDomainRepository.createDomain(inviteParam);
    }
```

```
package com.mszlu.xt.sso.domain.repository;

import com.mszlu.xt.model.params.InviteParam;
import com.mszlu.xt.sso.dao.InviteMapper;
import com.mszlu.xt.sso.dao.data.Invite;
import com.mszlu.xt.sso.domain.InviteDomain;
import org.springframework.stereotype.Component;

import javax.annotation.Resource;

@Component
public class InviteDomainRepository {
    @Resource
    private InviteMapper inviteMapper;

    public InviteDomain createDomain(InviteParam inviteParam) {
        return new InviteDomain(this,inviteParam);
    }

    public void save(Invite invite) {
```

```java
        inviteMapper.insert(invite);
    }
}
```

```java
package com.mszlu.xt.sso.domain;

import com.mszlu.xt.model.params.InviteParam;
import com.mszlu.xt.sso.dao.data.Invite;
import com.mszlu.xt.sso.domain.repository.InviteDomainRepository;

public class InviteDomain {

    private InviteDomainRepository inviteDomainRepository;
    private InviteParam inviteParam;

    public InviteDomain(InviteDomainRepository inviteDomainRepository,
InviteParam inviteParam) {
        this.inviteDomainRepository = inviteDomainRepository;
        this.inviteParam = inviteParam;
    }

    public void save(Invite invite) {
        inviteDomainRepository.save(invite);
    }
}
```

```java
package com.mszlu.xt.sso.dao;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.mszlu.xt.sso.dao.data.Invite;


public interface InviteMapper extends BaseMapper<Invite> {
}
```

5. 修改订单逻辑

> *只要用户提交订单，就将此订单算在推荐人身上，当用户购买完成，将邀请信息中的inviteStatus修改为1，代表订单已支付，邀请人有佣金*
>
> *作业：购买完成的状态更改，购买完成之后发送一条购买完成的消息mq当中，消费者中处理邀请信息的状态更改，不要增加购买流程的负担*

5.1 编码

```
 @PostMapping("submitOrder")
    public CallResult submitOrder(HttpServletRequest request,
@RequestBody OrderParam orderParam){
        orderParam.setRequest(request);
        return orderService.submitOrder(orderParam);
    }
```

OrderDomain 的submitOrder中：

```
 private void fillInvite(Long userId,String orderId) {
        HttpServletRequest request = this.orderParam.getRequest();
        Cookie[] cookies = request.getCookies();
        if(cookies == null){
                return;
        }
        List<Map<String,String>> billTypeList = new ArrayList<>();
        for (Cookie cookie : cookies) {
            String name = cookie.getName();
            String[] inviteCookie = name.split("_i_ga_b_");
            if (inviteCookie.length == 2){
                Map<String,String> map = new HashMap<>();
                map.put("billType",inviteCookie[1]);
                map.put("userId",cookie.getValue());
                billTypeList.add(map);
            }
        }
        for (Map<String,String> inviteMap : billTypeList) {
            //有推荐信息，构建邀请信息
            Invite invite = new Invite();
            invite.setInviteInfo(orderId);
            invite.setInviteStatus(0);
            invite.setInviteTime(System.currentTimeMillis());
            invite.setInviteType(InviteType.LOGIN.getCode());
            invite.setInviteUserId(userId);

 invite.setUserId(Long.parseLong(AESUtils.decrypt(inviteMap.get("userId"
))));
            invite.setBillType(inviteMap.get("billType"));
            invite.setCreateTime(System.currentTimeMillis());

 this.orderDomainRepository.createInviteDomain(null).save(invite);
        }
    }
```

OrderDomainRepository:

```
 @Autowired
    private InviteDomainRepository inviteDomainRepository;

    public InviteDomain createInviteDomain(InviteParam inviteParam) {

        return inviteDomainRepository.createDomain(inviteParam);
    }
```

```
package com.mszlu.xt.web.domain.repository;

import com.mszlu.xt.pojo.Invite;
import com.mszlu.xt.web.dao.InviteMapper;
import com.mszlu.xt.web.domain.InviteDomain;
import com.mszlu.xt.web.model.params.InviteParam;
import org.springframework.stereotype.Component;

import javax.annotation.Resource;

@Component
public class InviteDomainRepository {
    @Resource
    private InviteMapper inviteMapper;

    public InviteDomain createDomain(InviteParam inviteParam) {
        return new InviteDomain(this,inviteParam);
    }

    public void save(Invite invite) {
        inviteMapper.insert(invite);
```

```java
    }
}
```

```java
package com.mszlu.xt.web.domain;


import com.mszlu.xt.pojo.Invite;
import com.mszlu.xt.web.domain.repository.InviteDomainRepository;
import com.mszlu.xt.web.model.params.InviteParam;

public class InviteDomain {

    private InviteDomainRepository inviteDomainRepository;
    private InviteParam inviteParam;

    public InviteDomain(InviteDomainRepository inviteDomainRepository,
InviteParam inviteParam) {
        this.inviteDomainRepository = inviteDomainRepository;
        this.inviteParam = inviteParam;
    }

    public void save(Invite invite) {
        inviteDomainRepository.save(invite);
    }
}
```

```java
package com.mszlu.xt.web.dao;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.mszlu.xt.pojo.Invite;

public interface InviteMapper extends BaseMapper<Invite> {
}
```

作业

1. sso和web有重复代码
2. 通过发送mq的消息，统一在消费者进行处理