

尚筹网

[17-尚硅谷-尚筹网-前台-发起项目]

1 “发起项目”建模

1.1 创建数据库表

1.1.1 分类表

```
create table t_type
(
    id                int(11) not null auto_increment,
    name              varchar(255) comment '分类名称',
    remark            varchar(255) comment '分类介绍',
    primary key (id)
);
```

1.1.2 项目分类中间表

```
create table t_project_type
(
    id                int not null auto_increment,
    projectid         int(11),
    typeid            int(11),
    primary key (id)
);
```

1.1.3 标签表

```
create table t_tag
(
    id                int(11) not null auto_increment,
    pid               int(11),
    name              varchar(255),
    primary key (id)
);
```

1.1.4 项目标签中间表

```
create table t_project_tag
```

```
(
    id                int(11) not null auto_increment,
    projectid         int(11),
    tagid             int(11),
    primary key (id)
);
```

1.1.5 项目表

```
create table t_project
(
    id                int(11) not null auto_increment,
    project_name      varchar(255) comment '项目名称',
    project_description varchar(255) comment '项目描述',
    money            bigint(11) comment '筹集金额',
    day              int(11) comment '筹集天数',
    status            int(4) comment '0-即将开始, 1-众筹中, 2-众筹成功, 3-众筹失败',
    deploydate        varchar(10) comment '项目发起时间',
    supportmoney      bigint(11) comment '已筹集到的金额',
    supporter         int(11) comment '支持人数',
    completion        int(3) comment '百分比完成度',
    memberid          int(11) comment '发起人的会员 id',
    createdate        varchar(19) comment '项目创建时间',
    follower          int(11) comment '关注人数',
    header_picture_path varchar(255) comment '头图路径',
    primary key (id)
);
```

1.1.6 项目表项目详情图片表

```
create table t_project_item_pic
(
    id                int(11) not null auto_increment,
    projectid         int(11),
    item_pic_path     varchar(255),
    primary key (id)
);
```

1.1.7 项目发起人信息表

```
create table t_member_launch_info
```

```
(
    id                int(11) not null auto_increment,
    memberid          int(11)          comment '会员 id',
    description_simple varchar(255)    comment '简单介绍',
    description_detail varchar(255)    comment '详细介绍',
    phone_num          varchar(255)    comment '联系电话',
    service_num        varchar(255)    comment '客服电话',
    primary key (id)
);
```

1.1.8 回报信息表

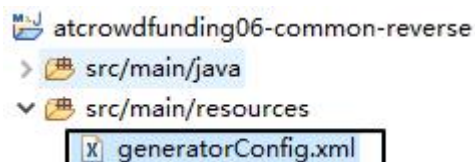
```
create table t_return
(
    id                int(11) not null auto_increment,
    projectid         int(11),
    type              int(4) comment '0 - 实物回报, 1 虚拟物品回报',
    supportmoney       int(11) comment '支持金额',
    content            varchar(255) comment '回报内容',
    count             int(11) comment '回报产品限额, “0” 为不限回报数量',
    signalpurchase     int(11) comment '是否设置单笔限购',
    purchase           int(11) comment '具体限购数量',
    freight            int(11) comment '运费, “0” 为包邮',
    invoice            int(4) comment '0 - 不开发票, 1 - 开发票',
    returndate         int(11) comment '项目结束后多少天向支持者发送回报',
    describ_pic_path   varchar(255) comment '说明图片路径',
    primary key (id)
);
```

1.1.9 发起人确认信息表

```
create table t_member_confirm_info
(
    id                int(11) not null auto_increment,
    memberid          int(11) comment '会员 id',
    paynum             varchar(200) comment '易付宝企业账号',
    cardnum            varchar(200) comment '法人身份证号',
    primary key (id)
);
```

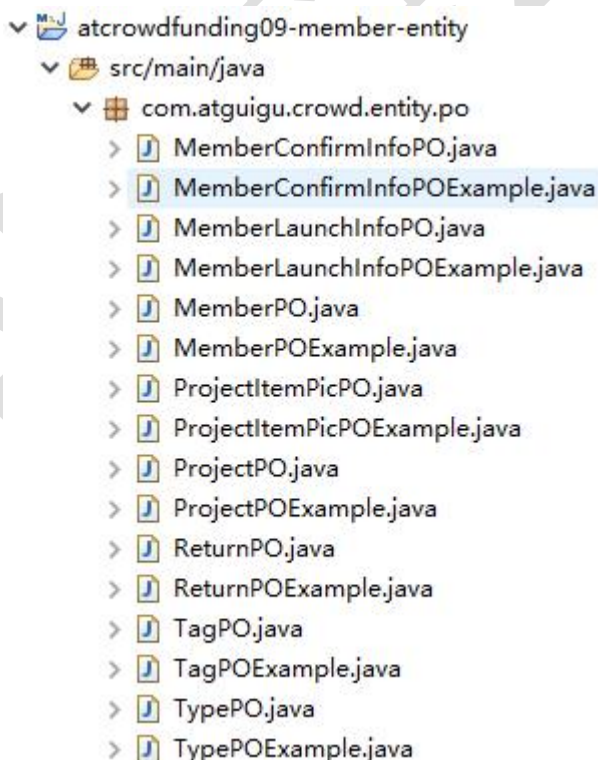
1.2 逆向工程

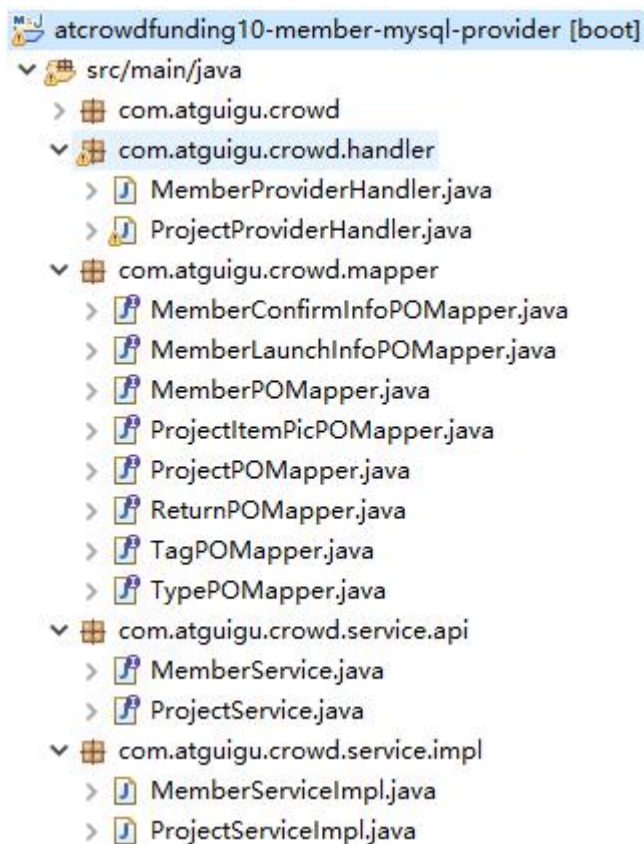
1.2.1 设置



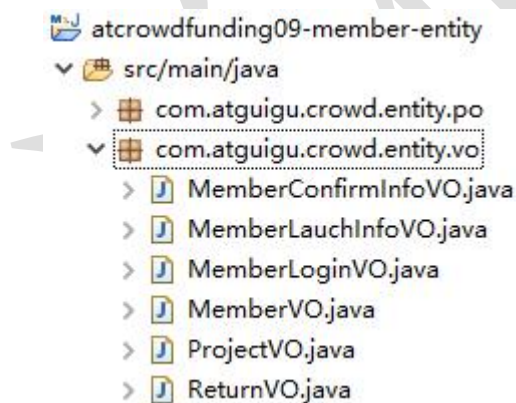
```
<!-- 数据库表名字和我们的 entity 类对应的映射指定 -->
<table tableName="t_type" domainObjectName="TypePO" />
<table tableName="t_tag" domainObjectName="TagPO" />
<table tableName="t_project" domainObjectName="ProjectPO" />
<table tableName="t_project_item_pic" domainObjectName="ProjectItemPicPO" />
<table tableName="t_member_launch_info" domainObjectName="MemberLaunchInfoPO" />
<table tableName="t_return" domainObjectName="ReturnPO" />
<table tableName="t_member_confirm_info" domainObjectName="MemberConfirmInfoPO" />
```

1.2.2 效果





1.3 创建 VO 对象



1.3.1 ProjectVO

```
@Data
@NoArgsConstructor
@AllArgsConstructor
public class ProjectVO implements Serializable {

    private static final long serialVersionUID = 1L;
    // 分类 id 集合
```

```
private List<Integer> typeIdList;

// 标签 id 集合
private List<Integer> tagIdList;

// 项目名称
private String projectName;

// 项目描述
private String projectDescription;

// 计划筹集的金额
private Integer money;

// 筹集资金的天数
private Integer day;

// 创建项目的日期
private String createDate;

// 头图的路径
private String headerPicturePath;

// 详情图片的路径
private List<String> detailPicturePathList;

// 发起人信息
private MemberLaunchInfoVO memberLaunchInfoVO;

// 回报信息集合
private List<ReturnVO> returnVOList;

// 发起人确认信息
private MemberConfirmInfoVO memberConfirmInfoVO;
}
```

1.3.2 MemberLaunchInfoVO

```
@Data
@NoArgsConstructor
@AllArgsConstructor
public class MemberLaunchInfoVO implements Serializable {
```

```
private static final long serialVersionUID = 1L;
// 简单介绍
private String descriptionSimple;

// 详细介绍
private String descriptionDetail;

// 联系电话
private String phoneNum;

// 客服电话
private String serviceNum;
}
```

1.3.3 ReturnVO

```
@Data
@NoArgsConstructor
@AllArgsConstructor
public class ReturnVO implements Serializable {

    private static final long serialVersionUID = 1L;

    // 回报类型：0 - 实物回报， 1 虚拟物品回报
    private Integer type;

    // 支持金额
    private Integer supportmoney;

    // 回报内容介绍
    private String content;

    // 总回报数量，0 为不限制
    private Integer count;

    // 是否限制单笔购买数量，0 表示不限购，1 表示限购
    private Integer signalpurchase;

    // 如果单笔限购，那么具体的限购数量
    private Integer purchase;

    // 运费，“0”为包邮
}
```

```
private Integer freight;

// 是否开发票，0 - 不开票， 1 - 开发票
private Integer invoice;

// 众筹结束后返回回报物品天数
private Integer returndate;

// 说明图片路径
private String describPicPath;

}
```

1.3.4 MemberConfirmInfoVO

```
@Data
@NoArgsConstructor
@AllArgsConstructor
public class MemberConfirmInfoVO implements Serializable {

    private static final long serialVersionUID = 1L;

    // 易付宝账号
    private String paynum;

    // 法人身份证号
    private String cardnum;

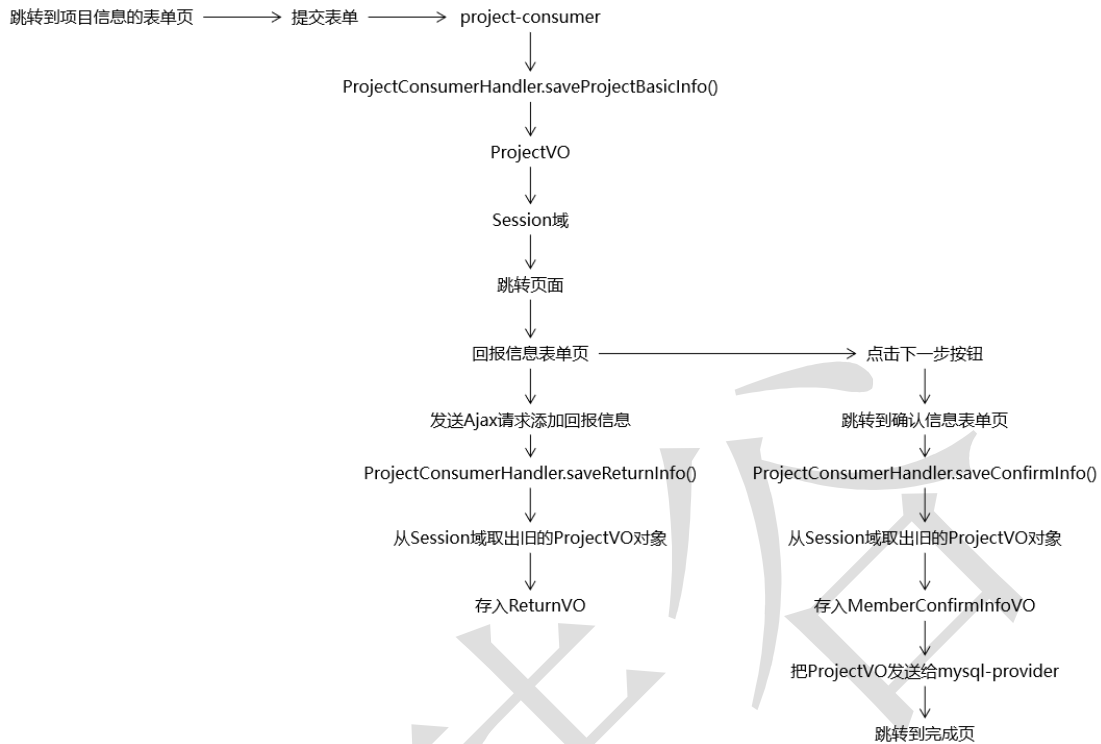
}
```

2 发起项目

2.1 总目标

将各个表单页面提交的数据汇总到一起保存到数据库。

2.2 思路



2.3 代码：跳转页面

2.3.1 配置访问 project-consumer 工程的路由规则



2.3.2 在 project-consumer 工程配置 view-controller



```
@Configuration
public class CrowdWebMvcConfig implements WebMvcConfigurer {

    @Override
    public void addViewControllers(ViewControllerRegistry registry) {

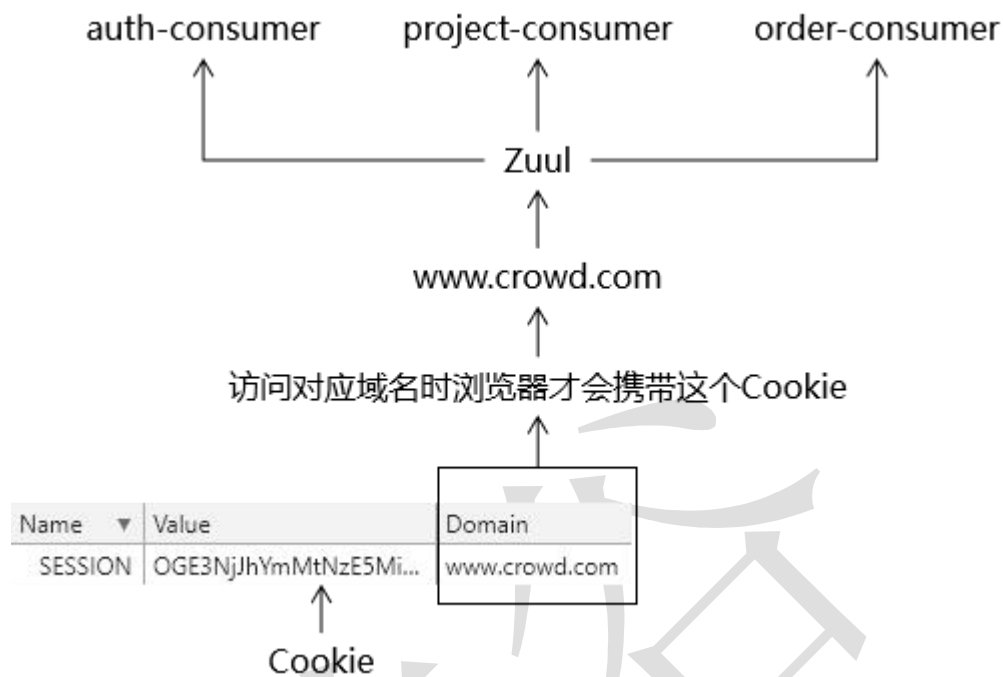
        // view-controller 是在 project-consumer 内部定义的，所以这里是一个不经过 Zuul
        // 访问的地址，所以这个路径前面不加路由规则中定义的前缀：“/project”
        registry.addViewController("/agree/protocol/page").setViewName("project-agree");
        registry.addViewController("/launch/project/page").setViewName("project-launch");

    }
}
```

2.3.3 页面上写地址需要注意

需要注意：前面要写上域名（如果没有配置域名写 localhost 一样），确保通过 Zuul 访问具体功能。

因为必须通过 Zuul 访问具体功能才能够保持 Cookie，进而保持 Session 一致。

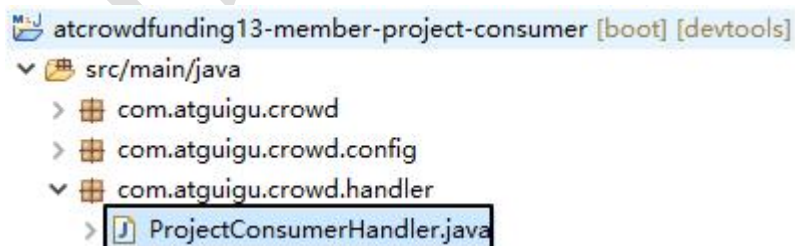


例如：退出系统链接



```
<a href="http://www.crowd.com/auth/member/logout">退出系统</a>
```

2.3.4 代码：接收表单数据



```
@RequestMapping("/create/project/information")
public String saveProjectBasicInfo(

    // 接收除了上传图片之外的其他普通数据
    ProjectVO projectVO,
```

```
// 接收上传的头图
MultipartFile headerPicture,

// 接收上传的详情图片
List<MultipartFile> detailPictureList,

// 用来将收集了一部分数据的 ProjectVO 对象存入 Session 域
HttpSession session,

// 用来在当前操作失败后返回上一个表单页面时携带提示信息
ModelMap modelMap
) throws IOException {

// 一、完成头图上传
// 1.获取当前 headerPicture 对象是否为空
boolean headerPictureIsEmpty = headerPicture.isEmpty();

if(headerPictureIsEmpty) {

    // 2.如果没有上传头图则返回到表单页面并显示错误消息
    modelMap.addAttribute(CrowdConstant.ATTR_NAME_MESSAGE,
CrowdConstant.MESSAGE_HEADER_PIC_EMPTY);

    return "project-launch";

}

// 3.如果用户确实上传了有内容的文件，则执行上传
ResultEntity<String> uploadHeaderPicResultEntity = CrowdUtil.uploadFileToOss(
    ossProperties.getEndPoint(),
    ossProperties.getAccessKeyId(),
    ossProperties.getAccessKeySecret(),
    headerPicture.getInputStream(),
    ossProperties.getBucketName(),
    ossProperties.getBucketDomain(),
    headerPicture.getOriginalFilename());

String result = uploadHeaderPicResultEntity.getResult();

// 4.判断头图是否上传成功
if(ResultEntity.SUCCESS.equals(result)) {
```

```
// 5.如果成功则从返回的数据中获取图片访问路径
String headerPicturePath = uploadHeaderPicResultEntity.getData();

// 6.存入 ProjectVO 对象中
projectVO.setHeaderPicturePath(headerPicturePath);
} else {

    // 7.如果上传失败则返回到表单页面并显示错误消息
    modelMap.addAttribute(CrowdConstant.ATTR_NAME_MESSAGE,
CrowdConstant.MESSAGE_HEADER_PIC_UPLOAD_FAILED);

    return "project-launch";
}

// 二、上传详情图片
// 1.创建一个用来存放详情图片路径的集合
List<String> detailPicturePathList = new ArrayList<String>();

// 2.检查 detailPictureList 是否有效
if(detailPictureList == null || detailPictureList.size() == 0) {
    modelMap.addAttribute(CrowdConstant.ATTR_NAME_MESSAGE,
CrowdConstant.MESSAGE_DETAIL_PIC_EMPTY);

    return "project-launch";
}

// 3.遍历 detailPictureList 集合
for (MultipartFile detailPicture : detailPictureList) {

    // 4.当前 detailPicture 是否为空
    if(detailPicture.isEmpty()) {

        // 5.检测到详情图片中单个文件为空也是回去显示错误消息
        modelMap.addAttribute(CrowdConstant.ATTR_NAME_MESSAGE,
CrowdConstant.MESSAGE_DETAIL_PIC_EMPTY);

        return "project-launch";
    }

    // 6.执行上传
    ResultEntity<String> detailUploadResultEntity = CrowdUtil.uploadFileToOss(
```

```
        ossProperties.getEndPoint(),
        ossProperties.getAccessKeyId(),
        ossProperties.getAccessKeySecret(),
        detailPicture.getInputStream(),
        ossProperties.getBucketName(),
        ossProperties.getBucketDomain(),
        detailPicture.getOriginalFilename());

// 7.检查上传结果
String detailUploadResult = detailUploadResultEntity.getResult();

if(ResultEntity.SUCCESS.equals(detailUploadResult)) {

    String detailPicturePath = detailUploadResultEntity.getData();

    // 8.收集刚刚上传的图片的访问路径
    detailPicturePathList.add(detailPicturePath);
} else {

    // 9.如果上传失败则返回到表单页面并显示错误消息
    modelMap.addAttribute(CrowdConstant.ATTR_NAME_MESSAGE,
        CrowdConstant.MESSAGE_DETAIL_PIC_UPLOAD_FAILED);

    return "project-launch";
}

}

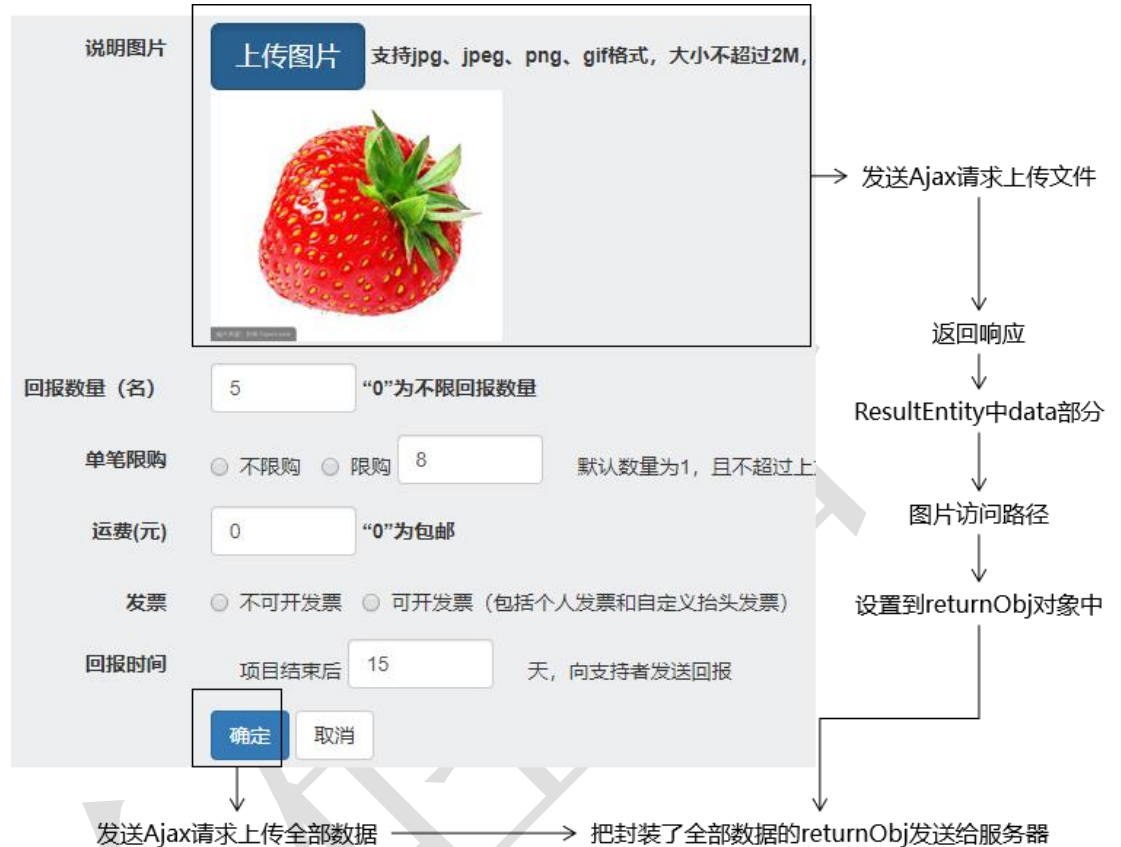
// 10.将存放了详情图片访问路径的集合存入 ProjectVO 中
projectVO.setDetailPicturePathList(detailPicturePathList);

// 三、后续操作
// 1.将 ProjectVO 对象存入 Session 域
session.setAttribute(CrowdConstant.ATTR_NAME_TEMPLATE_PROJECT, projectVO);

// 2.以完整的访问路径前往下一个收集回报信息的页面
return "redirect:http://www.crowd.com/project/return/info/page";
}
```

2.4 代码：收集回报信息

2.4.1 注意：上传图片 and 提交表单分开



3 收集回报信息

3.1 操作 1：接收页面异步上传的图片

```
atcrowdfunding13-member-project-consumer [boot]
├── src/main/java
│   ├── com.atguigu.crowd
│   ├── com.atguigu.crowd.config
│   └── com.atguigu.crowd.handler
│       └── ProjectConsumerHandler.java
```

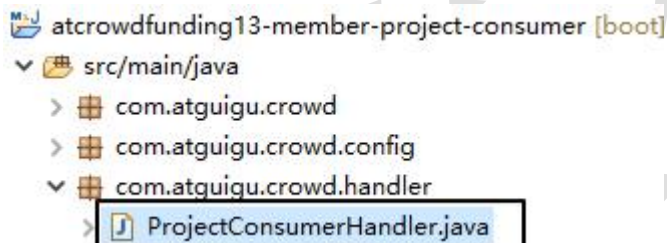
```
@RequestMapping("/create/upload/return/picture.json")
public ResultEntity<String> uploadReturnPicture(

    // 接收用户上传的文件
    @RequestParam("returnPicture") MultipartFile returnPicture) throws IOException {
```

```
// 1.执行文件上传
ResultEntity<String> uploadReturnPicResultEntity = CrowdUtil.uploadFileToOss(
    ossProperties.getEndPoint(),
    ossProperties.getAccessKeyId(),
    ossProperties.getAccessKeySecret(),
    returnPicture.getInputStream(),
    ossProperties.getBucketName(),
    ossProperties.getBucketDomain(),
    returnPicture.getOriginalFilename());

// 2.返回上传的结果
return uploadReturnPicResultEntity;
}
```

3.2 操作 2：接收整个回报信息数据



atcrowdfunding13-member-project-consumer [boot]

- src/main/java
 - com.atguigu.crowd
 - com.atguigu.crowd.config
 - com.atguigu.crowd.handler
 - ProjectConsumerHandler.java**

```
@ResponseBody
@RequestMapping("/create/save/return.json")
public ResultEntity<String> saveReturn(ReturnVO returnVO, HttpSession session) {

    try {
        // 1.从 session 域中读取之前缓存的 ProjectVO 对象
        ProjectVO projectVO = (ProjectVO)
        session.getAttribute(CrowdConstant.ATTR_NAME_TEMPLATE_PROJECT);

        // 2.判断 projectVO 是否为 null
        if(projectVO == null) {
            return ResultEntity.failed(CrowdConstant.MESSAGE_TEMPLATE_PROJECT_MISSING);
        }

        // 3.从 projectVO 对象中获取存储回报信息的集合
        List<ReturnVO> returnVOList = projectVO.getReturnVOList();

        // 4.判断 returnVOList 集合是否有效
        if(returnVOList == null || returnVOList.size() == 0) {
```



```
// 5.创建集合对象对 returnVOList 进行初始化
returnVOList = new ArrayList<>();
// 6.为了让以后能够正常使用这个集合，设置到 projectVO 对象中
projectVO.setReturnVOList(returnVOList);
}

// 7.将收集了表单数据的 returnVO 对象存入集合
returnVOList.add(returnVO);

// 8.把数据有变化的 ProjectVO 对象重新存入 Session 域，以确保新的数据最终能够
存入 Redis
session.setAttribute(CrowdConstant.ATTR_NAME_TEMPLATE_PROJECT, projectVO);

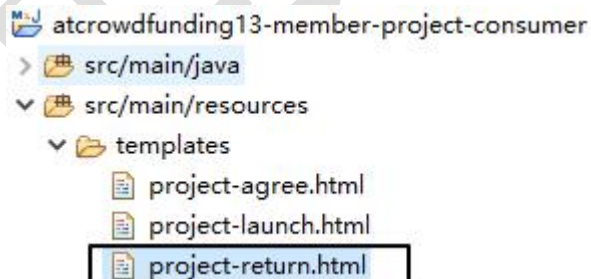
// 9.所有操作成功完成返回成功
return ResultEntity.successWithoutData();
} catch (Exception e) {
    e.printStackTrace();

    return ResultEntity.failed(e.getMessage());
}
}
```

3.3 跳转页面

从收集回报信息页面跳转到确认信息页面

3.3.1 页面上修改“下一步”按钮



```
<a th:href="@{/project/create/confirm/page.html}" class="btn btn-warning btn-lg">下一步</a>
```

3.3.2 添加 view-controller



```
registry.addViewController("/create/confirm/page.html").setViewName("project-confirm");
```

3.3.3 调整 project-confirm.html 页面



4 收集确认信息

4.1 点击提交按钮提交表单

4.1.1 修改提交按钮的 HTML 标签



```
<button type="button" id="submitBtn" class="btn btn-warning btn-lg">提交</button>
```

4.1.2 调整表单代码

```
<form id="confirmFomr" th:action="@{/project/create/confirm}" method="post" role="form">
    <div class="form-group">
        <label for="exampleInputEmail1">易付宝企业账号：</label><input type="email"
class="form-control" id="exampleInputEmail1" />
    </div>
    <div class="form-group">
```

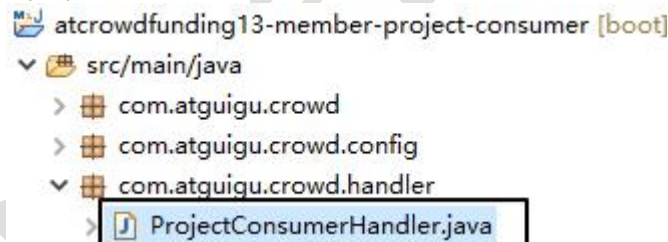
```
<label for="exampleInputPassword1"> 法 人 身 份 证 号 : </label><input  
type="password" class="form-control" id="exampleInputPassword1" />  
</div>  
</form>
```

4.1.3 给提交按钮绑定单击响应函数

```
<script type="text/javascript">  
$(function(){  
    $("#submitBtn").click(function(){  
        $("#confirmFomr").submit();  
    });  
});  
</script>
```

4.2 收集表单数据执行保存

4.2.1 project-consumer



```
@RequestMapping("/create/confirm")  
public String saveConfirm(ModelMap modelMap, HttpSession session, MemberConfirmInfoVO  
memberConfirmInfoVO) {  
  
    // 1.从 Session 域读取之前临时存储的 ProjectVO 对象  
    ProjectVO projectVO = (ProjectVO)  
session.getAttribute(CrowdConstant.ATTR_NAME_TEMPLATE_PROJECT);  
  
    // 2.如果 projectVO 为 null  
    if(projectVO == null) {  
        throw new RuntimeException(CrowdConstant.MESSAGE_TEMPLATE_PROJECT_MISSING);  
    }  
  
    // 3.将确认信息数据设置到 projectVO 对象中  
    projectVO.setMemberConfirmInfoVO(memberConfirmInfoVO);  
  
    // 4.从 Session 域读取当前登录的用户
```

```

        MemberLoginVO memberLoginVO = (MemberLoginVO)
session.getAttribute(CrowdConstant.ATTR_NAME_LOGIN_MEMBER);

        Integer memberId = memberLoginVO.getId();

        // 5.调用远程方法保存 projectVO 对象
        ResultEntity<String> saveResultEntity =
mysqlRemoteService.saveProjectVORemote(projectVO, memberId);

        // 6.判断远程的保存操作是否成功
        String result = saveResultEntity.getResult();
        if(ResultEntity.FAILED.equals(result)) {

            modelMap.addAttribute(CrowdConstant.ATTR_NAME_MESSAGE,
saveResultEntity.getMessage());

            return "project-confirm";
        }

        // 7.将临时的 ProjectVO 对象从 Session 域移除
        session.removeAttribute(CrowdConstant.ATTR_NAME_TEMPLATE_PROJECT);

        // 8.如果远程保存成功则跳转到最终完成页面
        return "redirect:http://www.crowd.com/project/create/success";
    }

```

4.2.2 声明 mysql-provider 的 Feign 接口

```

atcrowdfunding17-member-api [boot]
├── src/main/java
│   └── com.atguigu.crowd.api
│       ├── MySQLRemoteService.java
│       └── RedisRemoteService.java
└── src/main/resources

```

```

@FeignClient("atguigu-crowd-mysql")
public interface MySQLRemoteService {

    @RequestMapping("/get/memberpo/by/login/acct/remote")
    ResultEntity<MemberPO> getMemberPOByLoginAcctRemote(@RequestParam("loginacct")
String loginacct);

    @RequestMapping("/save/member/remote")

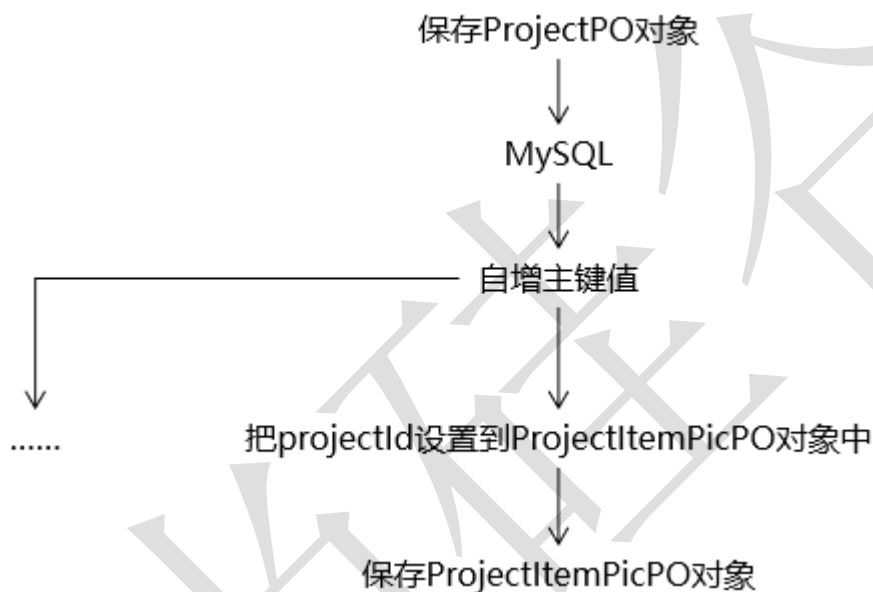
```

```
public ResultEntity<String> saveMember(@RequestBody MemberPO memberPO);

@RequestMapping("/save/project/vo/remote")
ResultEntity<String> saveProjectVORemote(@RequestBody ProjectVO projectVO,
    @RequestParam("memberId") Integer memberId);
}
```

4.3 执行数据库保存

4.3.1 需要在保存 ProjectPO 之后获取自增主键值



4.3.2 在 mysql-provider 中执行具体操作

```
atcrowdfunding10-member-mysql-provider [boot]
├── src/main/java
│   ├── com.atguigu.crowd
│   └── com.atguigu.crowd.handler
│       ├── MemberProviderHandler.java
│       └── ProjectProviderHandler.java
```

```
@RequestMapping("/save/project/vo/remote")
public ResultEntity<String> saveProjectVORemote(
    @RequestBody ProjectVO projectVO,
    @RequestParam("memberId") Integer memberId) {

    try {
        // 调用“本地”Service 执行保存
    }
}
```

```
projectService.saveProject(projectVO, memberId);

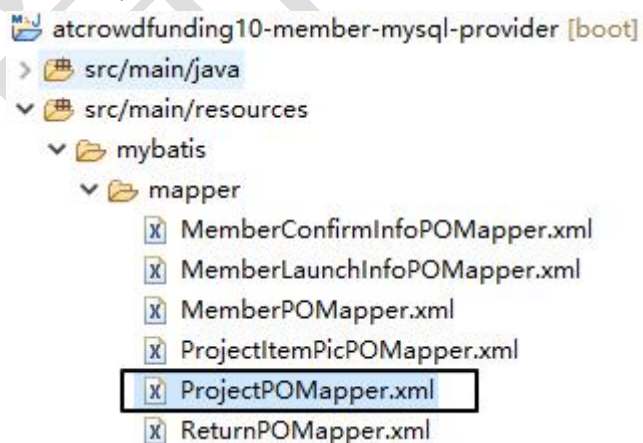
return ResultEntity.successWithoutData();
} catch (Exception e) {
    e.printStackTrace();

    return ResultEntity.failed(e.getMessage());
}
}
```

4.3.3 在 mysql-provider 的 Service 方法中执行保存

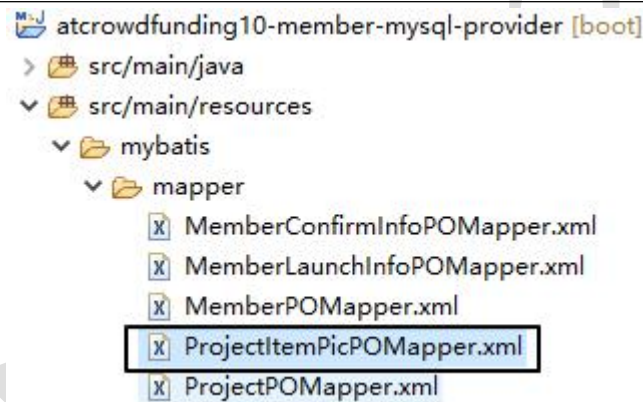


4.3.4 相关 SQL

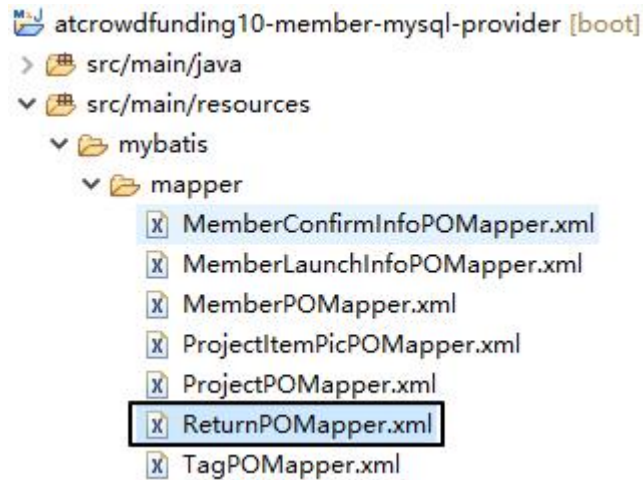


```
<!--
void insertTypeRelationship(
    @Param("typeIdList") List<Integer> typeIdList,
    @Param("projectId") Integer projectId);
```

```
-->
<insert id="insertTypeRelationship">
    insert into t_project_type(`projectid`,`typeid`) values
    <foreach collection="typeidList" item="typeid"
separator=",">#{projectId},#{typeid}</foreach>
</insert>
<!--
void insertTagRelationship(
    @Param("tagIdList") List<Integer> tagIdList,
    @Param("projectId") Integer projectId);
-->
<insert id="insertTagRelationship">
    insert into `t_project_tag`(`projectId`,`tagid`) values
    <foreach collection="tagIdList" item="tagId" separator=",">#{projectId},#{tagId}</foreach>
</insert>
```



```
<!--
void insertPathList(
    @Param("projectId") Integer projectId,
    @Param("detailPicturePathList") List<String> detailPicturePathList);
-->
<insert id="insertPathList">
    insert into t_project_item_pic (projectId, item_pic_path)
    valeus
    <foreach collection="detailPicturePathList" item="detailPath"
separator=",">#{projectId},#{detailPath}</foreach>
</insert>
```

```
<!--
void insertReturnPOBatch(
    @Param("returnPOList") List<ReturnPO> returnPOList,
    @Param("projectId") Integer projectId);
-->
<insert id="insertReturnPOBatch">
    insert into t_return (
        projectid,
        type,
        supportmoney,
        content,
        count,
        signalpurchase,
        purchase,
        freight,
        invoice,
        returndate,
        describ_pic_path
    )
    values
    <foreach collection="returnPOList" item="returnPO" separator=",">
        (
            #{projectId},
            #{returnPO.type},
            #{returnPO.supportmoney},
            #{returnPO.content},
            #{returnPO.count},
            #{returnPO.signalpurchase},
            #{returnPO.purchase},
            #{returnPO.freight},
```



```
        #{returnPO.invoice},  
        #{returnPO.returndate},  
        #{returnPO.describPicPath}  
    )  
</foreach>  
</insert>
```

4.3.5