

12 | 视频发布功能实现：怎样满足用户发布视频的需求？

2023-05-19 Barry 来自北京

《Python实战·从0到1搭建直播视频平台》



你好，我是 Barry。

我们都知道，在线视频平台除了满足用户在平台内浏览视频的需求之外，还需要满足用户发布视频的需求。

用户不单单是游客，还有可能成为平台的创作者，这样平台才能吸引更多的用户入驻。这节课的另一个核心知识点就是视频一键三连的功能实现，不难发现，这两块都属于用户与平台的交互功能。相信学完这节课以后，你的代码能力和需求设计能力会再一次强化。

shikey.com转载分享

接下来我们一起来看看该如何实现吧！

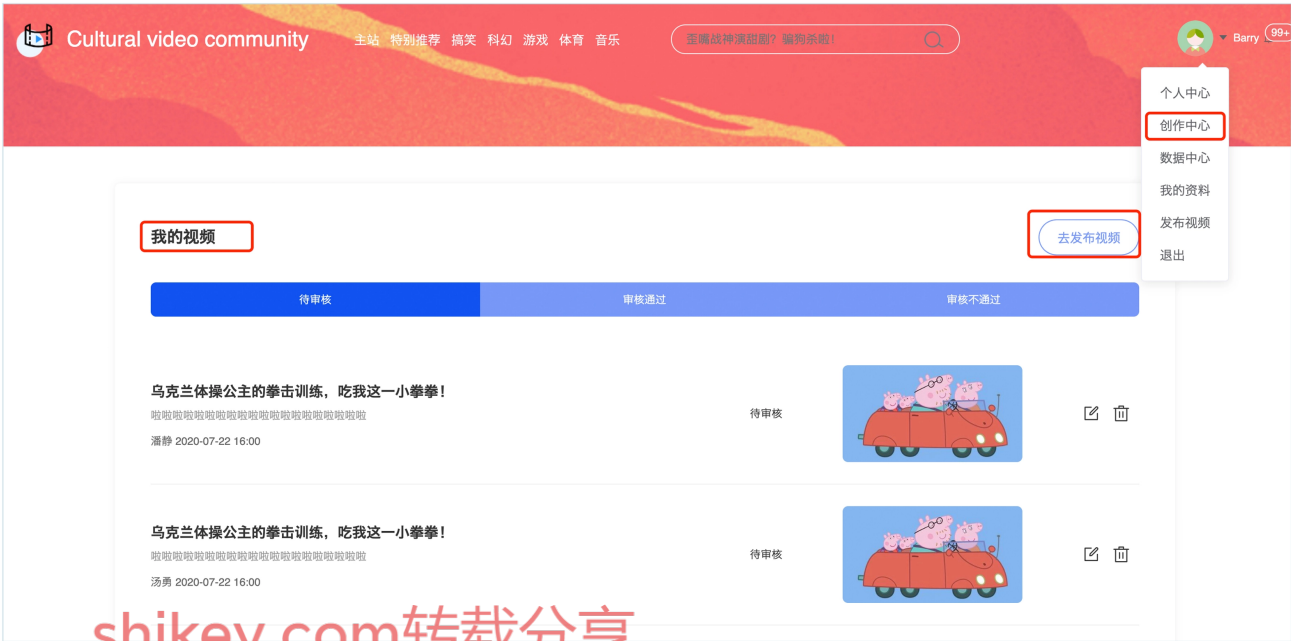
视频发布需求梳理

视频发布主要的使用者是平台的用户，核心就是满足用户在平台内上传视频、分享视频内容的需求。这里我们重点关注三个问题。

第一，发布视频的入口在哪里？我们选择在创作中心来完成视频内容的发布功能，创作中心主要用于用户的视频管理。

第二，发布视频都需要哪些信息？也就是在发布视频过程中，都需要用户上传哪些信息字段。同时，为了确保视频正常发布，平台还需要验证部分信息，例如封面上传只能上传图片，视频上传只能上传视频文件，这样可以更方便地把控内容。

第三，我们要模拟视频平台审核的功能。用户刚发布的视频，不能直接进入平台，还需要经过审核之后才能流入平台。这也是平台对视频内容监测和把控，确保平台内容的绿色健康。



前面这张图主要展示的就是平台的创作中心。

我们梳理一下用户的使用流程。创作中心的入口放在个人中心下面，用户点击自己的头像就能看到下拉列表。接着点击创作中心，就来到了创作中心页面，这时候用户能够看到“去发布视频”按钮，点击按钮就直接跳转到发布页面。最后用户输入视频相关信息、上传封面图片和视频文件，点击发布即可完成视频的发布动作，视频会进入待审核列表。

字段属性

我们结合视频发布页面的预览截图，反推一下我们需要的信息字段。

* 标题

最多输入20个文字/40个字符

封面

+

* 视频

点击上传

只能上传mp4/avi/mov/rmvb/flv文件，且不超过1G

描述

请输入视频描述

* 类别

请选择

标签

标签一 ×

标签二 ×

标签三 ×

+ New Tag

发布



其实这个生成过程应该是倒置的，也就是说我们要先考虑好用户需要上传哪些信息，然后根据功能的需求去设计页面。因为我已经完成了开发，我期望能够先给你看到效果，来引导你去思考，这样你以后碰到相关的功能需求就很容易处理了。

那用户发布视频到底需要哪些信息字段呢？这些字段的设计我们又该如何考虑？这绝对不是天马行空想出来的。

我们需要从展示和功能两个方向来考虑。

1. 展示维度可以帮助我们倒推所需字段。回想首页展示内容，我们在展示视频信息的时候需要视频的标题、封面、描述以及作者信息，所以平台需要用户在上传的时候就把这些数据提交给后台。

2. 从功能实现的方向看，就是确定发布视频的时候需要用户提供哪些字段，然后把字段存储在数据库，方便我们后期的查询或修改操作。

当然，所有的变量字段你都可以自己命名，但是一定要命名规范，然后要保持前后端字段统一。

字段名	含义	数据类型
title	标题	string
cover	封面	string
video_file	视频文件	string
desc	描述	string
type	类型	string
tag	标签	string



在明确了字段信息之后，就可以确定我们需要在视频上传时提交的信息。

页面布局与功能实现

创作中心的列表页的实现方式和个人中心里“我的关注”是一样的逻辑。

接下来我们重点看看视频发布页的布局 and 实现。这个页面是表单提交的形式，主要用到的组件就是 Element。

* 标题 最多输入20个文字/40个字符

封面

+

* 视频 点击上传

只能上传mp4/avi/mov/rmvb/flv文件，且不超过1G

描述 请输入视频描述

* 类别 请选择

标签 标签一 × 标签二 × 标签三 × + New Tag



其中带 * 号的表示必填项，我们只需要根据前面的功能需求来设置就可以。

这里我们先聚焦在视频发布页的表单实现上，至于更多表单属性的详细介绍，你可以课后查阅 [官网](#)，还是非常好理解的。

代码中的 onclick 方法都可以自定义，你不妨把我后面写的代码作为参考。如果你写的代码和我不完全一样也没关系，只要保证页面的功能完整就可以。

复制代码

```
1 <template>
2   <div class="article-edit">
3     <div class="content-main container">
4       <div class="content-wrapper">
5         <el-form
6           ref="form"
7           label-width="100px"
8           class="video-name"
9           :model="video"
10          :rules="rules"
11         >
12           <el-form-item label="标题" prop="title">
13             <el-input
14               type="text"
15             />
16           </el-form-item>
17         </el-form>
18       </div>
19     </div>
20   </div>
21 </template>
```

```
15         autocomplete="off"
16         placeholder="最多输入20个文字/40个字符"
17         v-model="video.title"
18         clearable
19     ></el-input>
20 </el-form-item>
21 <el-form-item label="封面" prop="cover">
22     <el-upload
23         class="avatar-uploader"
24         action="这里要写文件上传的地址"
25         :show-file-list="false"
26         :on-success="handleAvatarSuccess"
27         :before-upload="beforeAvatarUpload"
28     >
29         
30         <i v-else class="el-icon-plus avatar-uploader-icon"></i>
31     </el-upload>
32 </el-form-item>
33 <el-form-item label="视频" prop="video">
34     <el-upload
35         class="upload-demo"
36         action="这里写文件上传的地址"
37         :on-preview="handlePreview"
38         :before-upload="beforeAvatarUpload"
39         :on-remove="handleRemove"
40         :before-remove="beforeRemove"
41     >
42         <el-button size="small" type="primary">点击上传</el-button>
43         <div slot="tip" class="el-upload__tip">只能上传mp4/avi/mov/rmvpb/flv文
44     </el-upload>
45 </el-form-item>
46 <el-form-item label="描述" prop="desc">
47     <el-input
48         type="textarea"
49         :autosize="{ minRows: 2 }" //设置最小高度
50         placeholder="请输入视频描述"
51         v-model="video.desc"
52     ></el-input>
53 </el-form-item>
54 <el-form-item label="类别" prop="type">
55     <el-select v-model="video.type" placeholder="请选择">
56         <el-option
57             v-for="item in options"
58             :key="item.value"
59             :label="item.label"
60             :value="item.value"
61         ></el-option>
62     </el-select>
63 </el-form-item>
```

```

64     <el-form-item label="标签" prop="tag">
65         <el-tag
66             :key="tag"
67             v-for="tag in dynamicTags"
68             closable
69             :disable-transitions="false" //是否禁用渐变动画
70             @close="handleClose(tag)"
71         >{{tag}}</el-tag>
72         //以下方法主要用于新增标签
73         <el-input
74             class="input-new-tag"
75             v-if="inputVisible"
76             v-model="inputValue"
77             ref="saveTagInput"
78             size="small"
79             @keyup.enter.native="handleInputConfirm"
80             @blur="handleInputConfirm"
81         ></el-input>
82         <el-button v-else class="button-new-tag" size="small" @click="showInp
83     </el-form-item>
84     <button class="button-default" @click="onSave('form')">发布</button>
85 </el-form>
86 </div>
87 </div>
88 </div>
89 </template>

```

表单提交验证功能

Form 组件提供了表单验证的功能，我们只需要通过 `rules` 属性传入约定的验证规则，并且将 `Form-Item` 的 `prop` 属性设置为需校验的字段名即可。后面是具体的代码案例。

shikey.com转载分享

复制代码

```

1 //我们以标题为案例
2 //标题是必须让用户填写的一个字段，所以我们要通过表单验证，提示用户，如果没有填写不能提交
3 //同时我们要限制填入字段的长度
4
5 //表单内标题的代码
6
7 <el-form
8     ref="form"
9     label-width="100px"
10     class="video-name"
11     :model="video"
12     :rules="rules" //表单必须带上rules属性

```

shikey.com转载分享

```

13 >
14 <el-form-item label="标题" prop="title">
15   <el-input
16     type="age"
17     autocomplete="off" //原生属性，自动补全
18     placeholder="最多输入5个文字到10个字符" //输入框内显示输入提示
19     v-model="video.title" //绑定的字段
20     clearable
21   ></el-input>
22 </el-form-item>
23 //发布按钮
24 <button class="button-default" @click="onSave('form')">发布</button>
25 </el-form>
26 //在data内，我们需要写一个rules
27 rules: {
28   title: [
29     { required: true, message: "请输入标题", trigger: "blur" },
30     {
31       min: 5,
32       max: 10,
33       message: "最多输入5个文字到10个字符",
34       trigger: "blur",
35     },
36   ],
37 }
38
39 //到这里还不够，我们需要在methods中点击发布按钮时进行表单的验证
40 methods:{
41   onSave(formName) {
42     this.$refs[formName].validate((valid) => {
43       if (valid) {
44         //如果验证通过，直接在下面写提交的方法就可以
45       } else {
46         //如果验证不通过，则直接做出提示
47         return false;
48       }
49     });
50 }

```

shikey.com转载分享

shikey.com转载分享

以上就是完整的一个表单验证过程。你会发现使用组件很便捷，不需要像之前那样，通过单独写 Javascript 来实现。

这里补充一句，课程里我重点和你讲解的是 Element 组件库中不容易理解的部分，以及与我们项目实现更相关的组件。其他组件理解起来也不难，你可以课后对照官网做补充了解。

我们来看看实现后的效果。我们输入一些不满足要求的内容，点击提交按钮看一下页面的效果。



一旦输入内容超出规定范围，就会出现后面这样的提示。



文件上传功能

文件上传依然可以通过 Element 的组件去实现。为了让你在应用过程中少走弯路，我先带你梳理一下注意事项。

在项目中，通过封面照片上传功能代码案例，让你之后在应对各种系统的文件上传功能，你都可以通过下面的方法来实现。

首先，我们要了解 Upload 组件，它的作用是在前端实现文件以及图片的上传。

我们先来重点认识 Upload 组件的几个参数，表格里列出的参数是我们使用组件时常用到的方法，参数说明来自官网。

参数	说明	类型
action	必选参数，上传的地址	string
on-preview	点击文件列表中已上传的文件时的钩子	function(file)
on-remove	文件列表移除文件时的钩子	function(file, fileList)
on-success	文件上传成功时的钩子	function(response, file, fileList)
on-error	文件上传失败时的钩子	function(err, file, fileList)
on-progress	文件上传时的钩子	function(event, file, fileList)
before-upload	上传文件之前的钩子，参数为上传的文件，若返回 false 或者返回 Promise 且被 reject，则停止上传。	function(file)
before-remove	删除文件之前的钩子，参数为上传的文件和文件列表，若返回 false 或者返回 Promise 且被 reject，则停止删除。	function(file, fileList)



了解了常用参数，还需要结合具体实现代码才能加深理解。

复制代码

```
1 //表单中页面代码
2 <el-form-item label="封面" prop="cover">
3   <el-upload
4     class="avatar-uploader"
5     action="#" //这里要写服务器上传地址，前期没有可以直接写#号
6     :show-file-list="false"
7     :on-success="handleAvatarSuccess" // 上传成功之后的钩子方法
8     :before-upload="beforeAvatarUpload" //上传前的钩子方法
9   >
10     //上传成功之后展示图
11    <i v-else class="el-icon-plus avatar-uploader-icon"></i>
12  </el-upload>
13 </el-form-item>
```

以上就是组件的写法，结合注释还是很好理解的。你要关注的是钩子方法，我们需要在上传过程中针对不同情况灵活应用它。

我们再结合一个实用案例体会一下钩子方法的妙处。比方说，我们想在上传过程中判断文件格式，要怎么实现呢？


这里就要用到 `before-upload` 钩子，它的作用就是在文件上传之前判断文件类型。下面就是钩子方法，该方法有且仅有一个参数返回，就是 `file` 文件信息，我们结合代码实现具体看一下。

 复制代码

```
1 beforeAvatarUpload(file) {
2   //file中有文件的相关信息，你可以在控制台直接打印查看
3   //file.type是文件的类型
4   //file.size是文件的大小
5   const isJPG = file.type === "image/jpeg";
6   const isLt2M = file.size / 1024 / 1024 < 2;
7   if(!isJPG){
8     this.$message.error("上传头像图片只能是 JPG 格式!");
9   }else if(!isLt2M){
10    this.$message.error("上传头像图片大小不能超过 2MB!");
11  }else{
12    //当你没有文件上传服务器地址的时候，这里主要用于图片展示
13    this.video.cover = URL.createObjectURL(file);
14    return false
15  }
16  return isJPG && isLt2M;
17 }
```

shikey.com转载分享

最终文件上传到后端并实现存储，我们就直接使用 `axios` 来做接口调用，`POST` 方法传值。这里我们重点需要关注的是对于 `file` 文件的上传接口，我们需要使用 `FormData` 格式进行数据封装，提交方式是后面这样。

 复制代码

```
1 let formdata = new FormData()
2 formdata.append('title', this.video.title)
3 formdata.append('video_url', this.video.video_url)
4 formdata.append('cover', this.video.cover)
5 formdata.append('userId', this.userId)
```

```
6
7 this.$axios
8   .post('请求的接口地址', this.formdata)
9   .then((data) => {
10     if (data.code === 200) {
11       this.$message({
12         type: "success",
13         message: data.msg,
14       });
15       this.$router.push("/home/admin"); //提交成功之后回到创作中心
16     }
17   });
```

好，进行到这里，视频发布功能的设计与开发已经大功告成了！恭喜我们取得阶段性的成果。

如何修改发布视频内容

为了更灵活地管理发布的视频内容，给用户更好的使用体验，已发布视频内容的修改功能也是必不可少的。

我们同样先推演一下用户的操作流程。用户可以在创作中心的待审核列表、审核通过列表、审核不通过列表中，选择相应的视频，点击修改按钮，然后跳转到编辑页面就可以修改视频内容了。

那么修改视频的流程与发布视频有什么不同呢？

不同就是，用户点击修改按钮跳转到修改页面以后需要获取当前视频的信息，并不是像发布视频那样，表单是完全清空的状态。所以这时我们需要在页面加载的时候，直接把对应视频的信息显示出来，方便用户去修改表单。

那么我们下面要做的就是确定如何获取视频信息，有两种方式供我们选择。


第一种方法就是利用查询接口来查询。当用户点击修改按钮跳转到修改页面时，需要调用视频查询接口获取数据，接着赋值给表单。

第二种方法就是利用浏览器缓冲，这相当于把点击当前视频的信息直接存储在 localStorage 中，然后在修改页面的 mounted 方法中直接赋值。

比较一下这两个方法，我们会发现接口请求的方式更加得当。毕竟存储在缓存中，存在一定的安全隐患。


确定了接口请求的信息获取方法，我们来看看具体的实现代码。

我们在 home/admin.vue 文件中。我们先来看 HTML 部分，我给你写了详细的注释。

 复制代码

```
1 //HTML部分
2 <span>
3   <i class="el-icon-edit-outline" @click.stop="goEdit(item.articleId)"></i>
4   //修改按钮，触发方法goEdit，同时传递视频的ID
5 </span>
```

JS 部分我们要实现带着参数进行页面跳转，后面是具体的代码案例。


 复制代码

```
1 //Js部分
2 //在跳转路由中，可以直接以+id的这样的写法来传值，采用的是route的params方法，也可以多个值进行传
3 goEdit(id) {
4   this.$router.push("/video/edit/" + id);
5 }
```

shikey.com转载分享

在 edit.vue 文件中，我们要写出获取对应视频信息的方法。

shikey.com转载分享

 复制代码

```
1 mounted() {
2   this.video.userId = this.user.userId;
3   // 直接可以通过this.$route.params.id获取到传输的ID
4   // params 是route传参方式,
5   if (this.$route.params.id) {
6     this.$axios
7       .get(
```

```
8         "接口地址" +
9         this.$route.params.id
10    )
11    .then((data) => {
12        //this.video就是表单的数据
13        this.video = data.data;
14    });
15    }
16 }
```

到这里，我们就完成了已发布视频的修改功能，这样就能更灵活地管理视频发布内容了。

总结

我们回顾一下这节课的内容。这节课我们一起设计并实现了视频发布功能，这是视频平台里最核心的一个模块。

整个实现过程中，有三个核心知识需要我们重点关注。

第一，在功能开发前的设计环节中，一定要明确**页面呈现形式和对应的属性字段**。比如视频发布的字段，可以从功能开发和内容展示两个维度考虑，确保功能的完整性。

第二，组件的使用。这节课我们又接触到了新组件，灵活应用组件能够帮我们快速开发功能，提升工作效率。对于组件，我推荐的学习方法是根据 Element 官网的案例代码进行实践，只要你充分理解了组件的 API，应用起来就非常容易。

第三是视频修改功能的实现，其实它的实现逻辑和新增视频类似，只不过多了一步提前获取修改数据。确定视频信息获取方法的时候，我们对比了接口调用和浏览器缓冲这两种方式，选择了安全性更好的第一种方式。

shikey.com转载分享

这几节课代码实战内容比重比较大，希望你课后多多上手练习，有什么疑问可以在留言区和我交流。

下节课，我们将会完成个人中心的另一个核心模块——数据中心，敬请期待。

思考题

如果想在表单中 el-input 组件中实现输入数字的验证功能，你有什么好办法？

欢迎你在留言区和我交流讨论。如果觉得这节课的内容对你有启发，也推荐你把这节课分享给更多朋友。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

精选留言 (2)



peter

2023-05-20 来自北京

Q1: python开发网站的话，能处理高并发吗？（是不是只适合小型、并发不高的网站？）

Q2: “修改视频”，是修改视频信息吗？还是修改视频本身的内容？

作者回复: 1、Python可以很好地处理高并发，因为它本身就是一种高并发的编程语言，而且有很多用于处理并发的库和工具。像Flask框架，提供了很多用于处理并发的工具和库，如锁、队列、请求池等，这个是完全可以处理的。

2、主要是修改视频的信息，你可以完全更换视频的所有信息，常规的操作是替换其中的一些描述信息，这个功能的需求你可以在开发中自己定义也是OK的。



1



BigData ~ 兰兴星

2023-05-19 来自江苏

老师，用户发布视频后如何审核这能不能加一期学习一下

shikey.com转载分享

作者回复: 可以的，之后再合适的时间节点可以加一节，我们一起来交流学习一下视频的审核机制和实现方式，期望对你有所帮助，一起加油。

shikey.com转载分享



1