阿里云 开发者社区 | 高校计划 | 视觉智能开放平台

5天入门视觉AI

开启云上开发，两大实践场景助你入门视觉AI

阿里云高校计划视觉AI训练营必备教材

身份识别/电子相册入门速成，2020人工智能应用新姿势

藏经阁

阿里云开发者电子书系列

阿里云 开发者社区

阿里云开发者“藏经阁”
海量免费电子书下载

扫码加入高校计划

序言

高校计划简介

阿里云高校计划为在校生免费提供了 2.68 亿小时云服务器算力，云计算入门训练营、达摩院 AI 视觉训练营等实践资源以及按职业路径推荐的系统专业课程，点击高校计划立即申请。

阿里云视觉智能开放平台简介

阿里云视觉智能开放平台是基于阿里巴巴视觉智能技术实践经验，面向视觉智能技术的开发与应用用户，为其提供好用、易用、普惠的视觉智能 API 服务，帮助企业、开发者快速建立视觉智能技术的应用算法的综合性视觉 AI 算法平台。为了更好的帮助中小企业和独立开发者快速对接视觉 AI 算法，平台免费开放现有的 100 余种视觉 AI 算法服务的使用权限，服务调用不收取任何费用，立即前往体验！

目录

视觉生产技术探索和应用

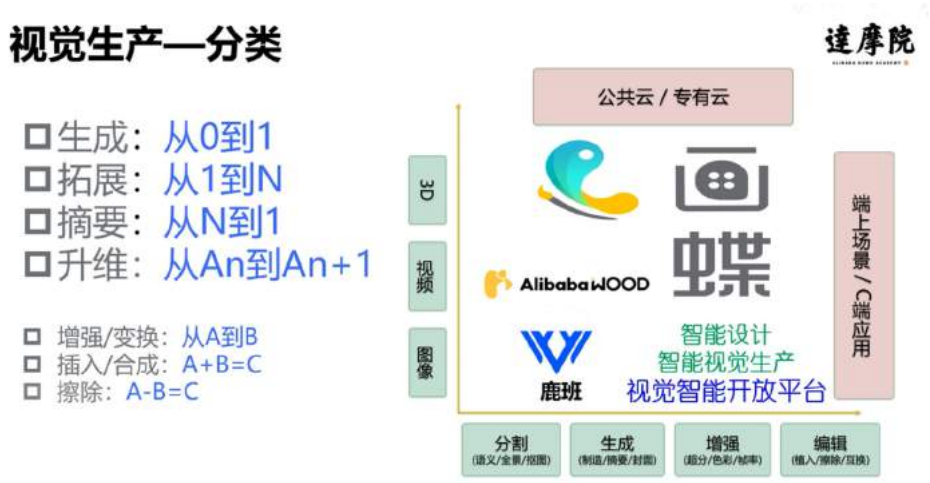
一、视觉生产简介与理解

（一）定义

就视觉而言一般来说有两大类：一类是视觉理解，比如检测、分割等；另外一类是视觉生产，也可以理解为怎么去产生视觉，指通过一个 / 一系列视觉过程，产出新的视觉表达。如下图所示，有两点需要注意，一是这里的视觉表达指的是人或机器能够感知的图像视频，而不是标签或特征，二是产出新的视觉表达，和输入是不一样的视觉表达。在过去，图中所示的过程大多数由人来完成，比如设计师、美工等用 PS 等工具完成，现在，我们希望通过技术能够实现这个过程。

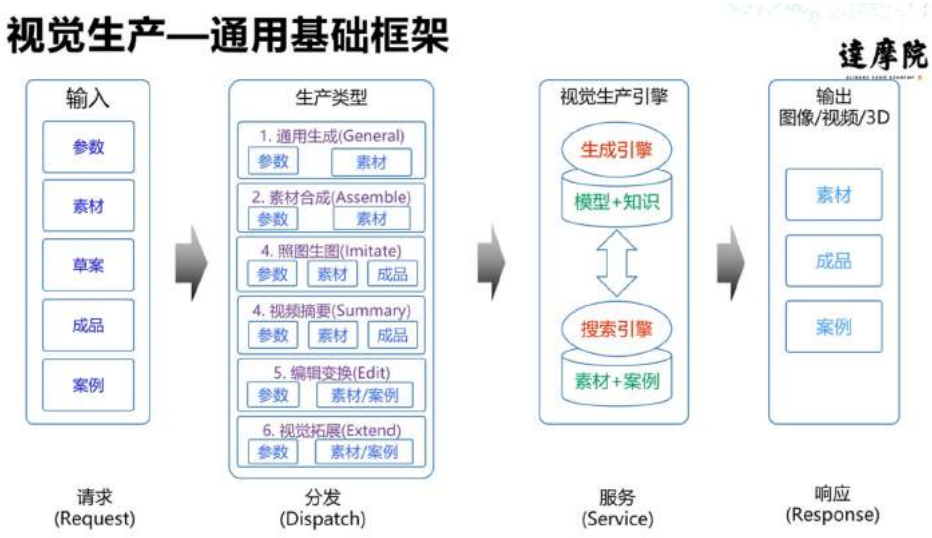
（二）分类

如下图所示，视觉生产主要包括生成、拓展、摘要、升维，另外还有增强 / 变换、插入 / 合成、擦除等。达摩院在该领域已经投入了很多人力和精力，也形成了一些产品，比如鹿班、画蝶、视觉智能开放平台等。



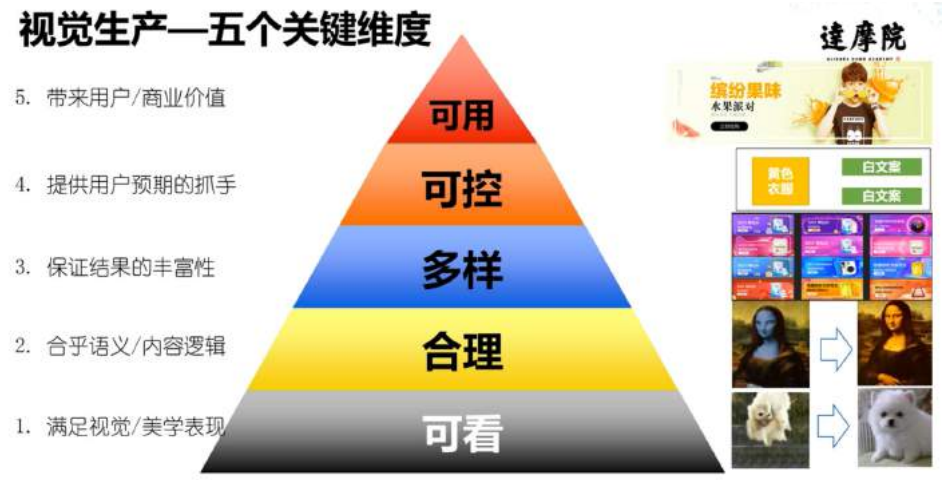
（三）通用基础框架

视觉生产有自己基本的通用框架，如下图所示。可能在细节上有细微不同，但是一般来讲其逻辑是类似的，包括请求（Request）、分发（Dispatch）、服务（Service）和响应（Response）四大部分。



（四）五个关键维度

如下图所示，要保证视觉生产有一个好的结果或者说可用的结果，其至少应满足可看、合理、多样、可控、可用五个维度，只有这样，才能在工业界产生真正的价值，而不仅仅是一个停留于理论的技术。



二、精细理解——寻微入里

如果想生产一个视觉，首先我们要理解输入的视觉，也就是需要精细的理解视觉。“理解”其实包括如下几个过程：

- **识别：**知道是什么，比如人的识别、物的识别；
- **检测：**识别 + 知道在哪，比如缺陷检测、多目标检测；
- **分割：**识别 + 检测 + 知道每一个像素是什么。

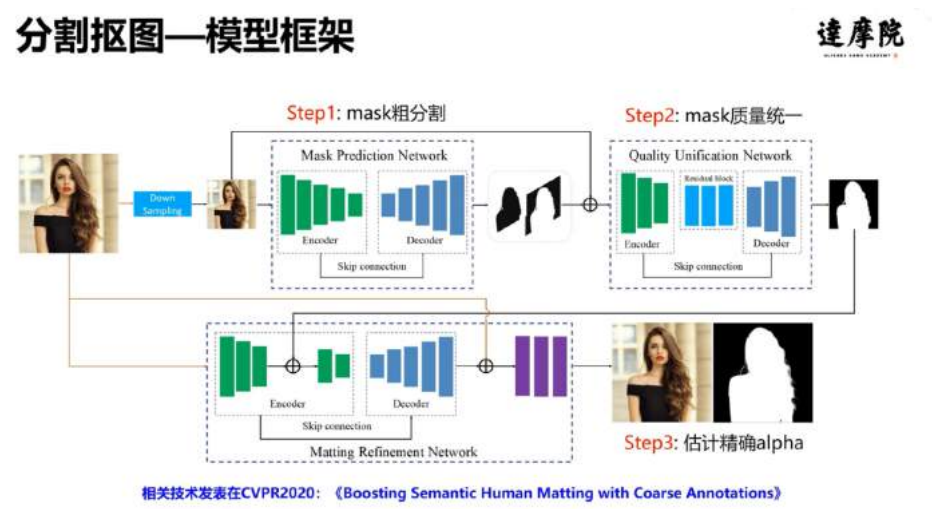
其中，视觉分割是生产的必要前置步骤，也是学术界和工业界的一个热点，同时也是难点，因为进行分割时往往有复杂的背景和各种遮挡关系，或者在分割时对其要求非常高，比如发丝级、镂空等，另外还可能面临边缘发色、透明材质、多目标 / 多尺度进行分割等问题。实际上，分割时遇到的这些难题归根到底是标注成本高、数据严重不足的问题，更进一步，即使标注出来了，但是想要精细得将其分割出来，成本

是成倍增加的。

（一）分割抠图解题思路

如下图所示，分割抠图包括不同的层次，从语义分割到实例分割再到 Image Matting。

整体来说分割抠图的过程是比较复杂的，总的来说我们的思路是先拆解再丰富数据样本，其框架如下图所示。



（二）分割抠图效果展示

如下图所示，通过上面的技术我们在发丝级的分割和镂空细节特殊场景等例子中取得了较好的效果。目前，在阿里巴巴内部分割抠图技术是使用最广的视觉 AI 技术。

基于分割抠图技术，我们可以对分割进行拓展，进行多式多样的分割，比如对人的分割，可以从图像中分割出人的头像、单独分割出头发、单独分割出人脸等，更进一步，除了静态图像的分割，我们甚至对视频进行分割，在动态的视频中提取人物。类似的，对于动物分割、车辆分割、商品分割、动画分割等，我们也可以进行相应的拓展，来丰富我们的分割粒度。另外的，对于场景抠图，比如天空分割，我们把天空分割出来的同时把人物、物体等分割出来，进行场景分割的拓展。

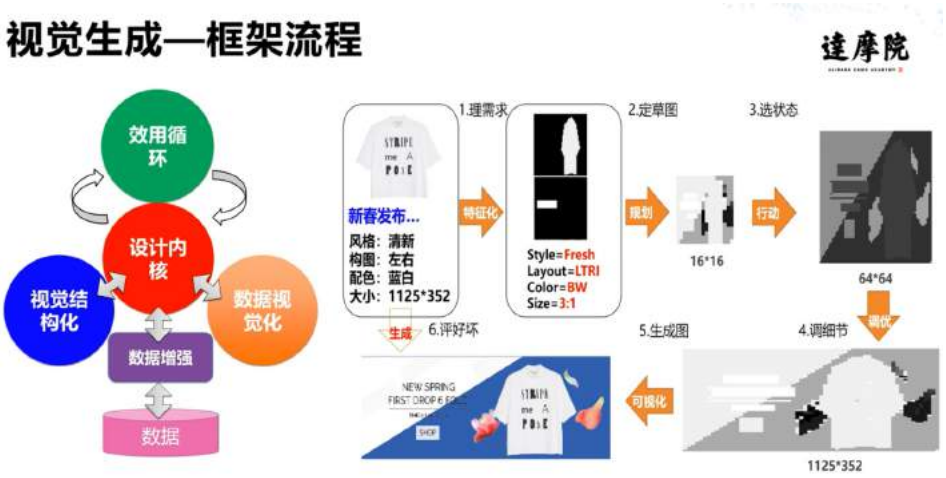
三、视觉生成——从无到有

在完成分割之后，我们对视觉有了精细的理解，才能够进行下一步的工作。

（一）视觉生成——鹿班

最早的时候我们做了产品鹿班。鹿班是视觉生成领域在业界落地的先行者，对外

提供大规模在线的 AI 设计服务。它是针对平面图像设计生成的产品，一开始在阿里巴巴内部大规模使用，目前已经通过阿里巴巴的云服务对外提供服务。鹿班的视觉生成的框架流程如下图所示，其大概过程包括理需求、定草图、选状态、调细节、生成图、评好坏 6 个步骤。



鹿班在多个领域有着广泛的应用，一开始是在电商领域，主要有以下两个能力：

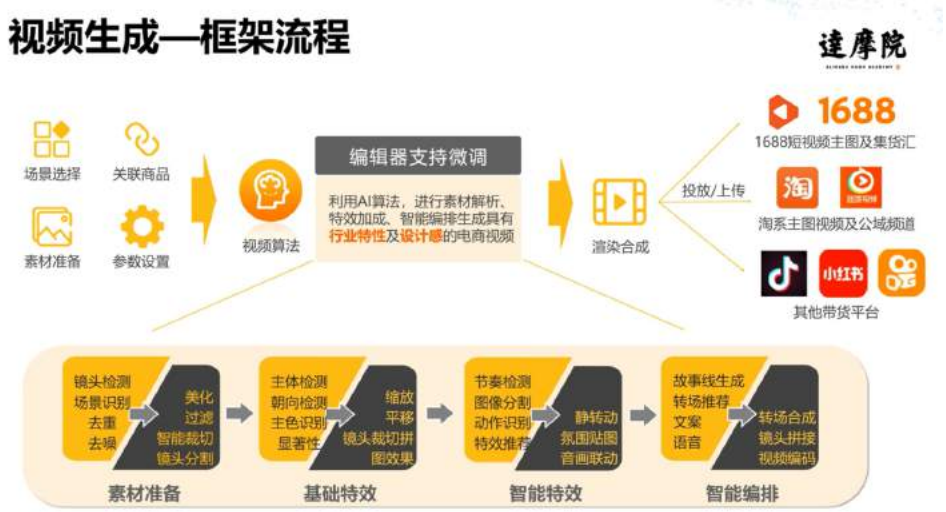
- **照图生图：**参考原图，将风格、布局等信息学习并迁移到目标数据上；
- **个性化设计：**多元化设计风格，结合商品品类、投放场景、目标客群的差异进行定制化设计。

如下图所示，鹿班还可以用来做场景智能美工，用 AI 实现场景设计能力，大大降低人力成本。

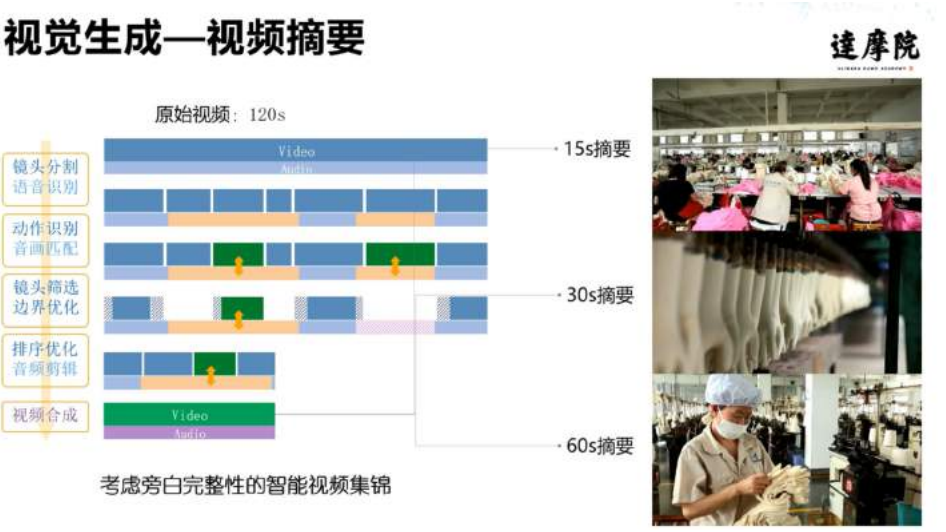
当然，鹿班的应用不仅是以上几个行业，它在各行各业都有着广泛的应用，并且在各个行业的应用都会产生不同的效果，依据场景来进行赋能。

（二）视觉生成——AlibabWood

鹿班主要针对的是平面图生成，但是实际上更多场景需要的是视频生成，比如当前流行的短视频，对此阿里巴巴也设计了 AlibabWood 这款产品，专注于短视频的生成，目前已经累计生成了超过 2000 万个短视频，同时还有剧本生成、智能文案生成、自动剪辑、智能音乐推荐等实用功能。AlibabWood 的框架流程如下图所示，总体包括素材准备、基础特效、智能特效和智能编排四大步骤，包括了众多技术在内。



AlibabWood 有着众多应用案例，比如场景化智能视频的生成，还可以规模化生成特效视频，当然，如下图所示，在生成了视频之后，如果有多个视频，还可以进行视频摘要的生成以及考虑旁白完整性的智能视频集锦。



视频封面的生成也是一个重要的应用，如下图所示，它可以对视频内容全自动完成质量审核、内容分析与图像增强，输出多帧静止或者动图，这个过程用到了图像增

强、内容分析等技术，也是在视频生成之后一个非常重要的技术应用。

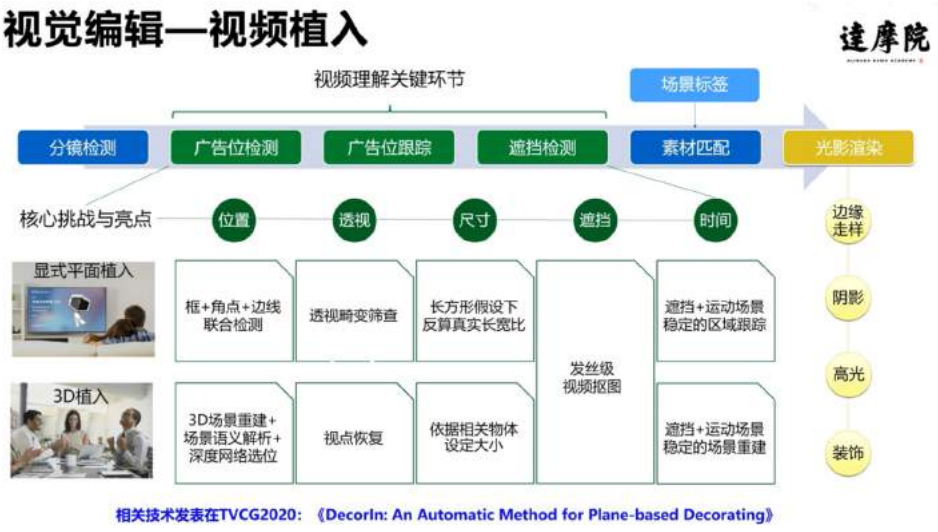
四、视觉编辑——移花接木

一个视频，能不能变成另外一个？这就要用到视频编辑技术，主要包括增删查改等功能。

（一）视频植入

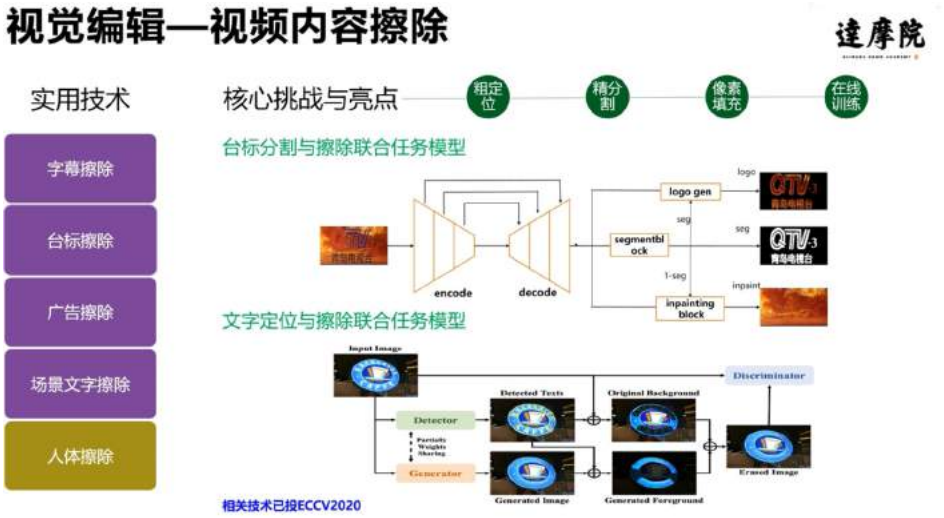
视频植入就是在视频中加入一些本来没有的内容，如下图所示，当前其应用最广泛的就是广告界。

如下图所示，视频植入是一项非常复杂的技术，需要考虑到方方面面，比如广告位检测、广告位跟踪等等，，有时会遇到遮挡、移出屏幕等复杂情况跟踪，而且在视频植入之后还要考虑广告是否能够跟视频细节匹配、光影渲染等问题。



（二）视觉内容擦除

上面视频植入是增加一些东西到视频中，有时候我们也要从视频中擦除一些东西，比如字母擦除、台标擦除、广告擦除等等，其核心挑战是分割，只有更精确的分割才能够精确的擦除。



（三）画幅变化

有些时候我们需要对视频进行修改，比如某段视频在拍摄的时候是在 4：3 的情况下进行的，在 ipad、PC、手机上面播放的时候出现了尺寸不匹配的情况，这时候就要进行画幅变化，变化之后为了有完整的视觉效果，需要进行内容补全，如下图所示。

（四）图像尺寸变化

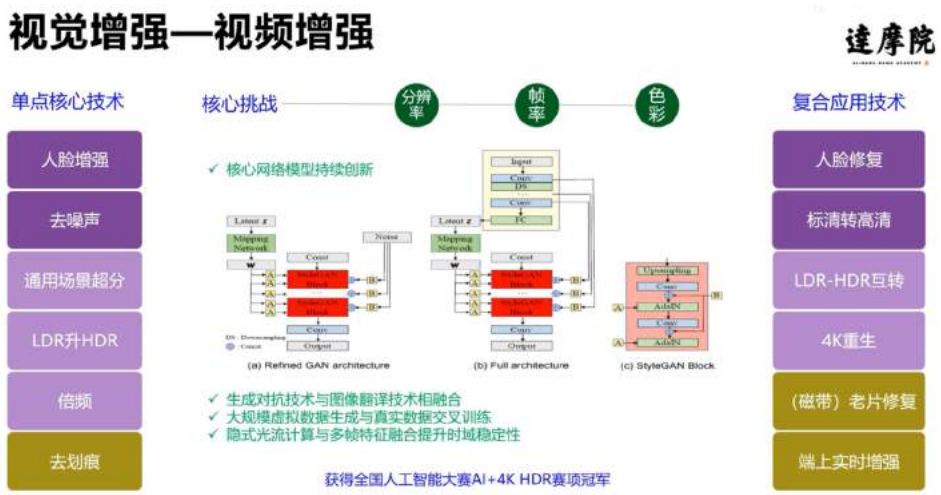
为了节省时间和精力，我们还可以进行图像尺寸的自动变化，这样子在某一个场景下设计的海报就可以更方便的用在其他场景中。

五、视觉增强——修旧如新

视觉增强就是对视频的内容进行一些改变，以达到视频某些方面的改善效果。

（一）视频增强

对视频效果的增强称为视频增强，如下图所示，它包括很多技术在内，包括单点核心技术和复合应用技术。



（一）视觉增强实例

1. 人脸修复

人脸是最重要的目标对象，对人像进行细节修复增强，有很重要的意义和价值，如下图所示，可以用视觉增强技术对人脸进行修复增强，突出主要信息。

2. 渲染图超分

CG 渲染时间几乎与图像分辨率成正比，高质量真实感渲染需要 30 分钟才能生产一张图像，针对 CG 渲染流水线研发的超分辨率技术可以把低分辨率图像放大到与高清原图一样的清晰度。

3. 视频超分

除了对图像进行超分外，我们还可以对视频进行超分，使得视频更加清晰，增加显示效果。

4. 视频插帧

对视频进行插帧，一般的视频可能感受不到效果，但是在运动的场景、网络不好的在线视频等场景，对视频进行插帧可以有效的减少视频的卡顿感。

5. HDR 色彩扩展

除了帧率之外，色彩也是一个很重要的元素，也是视频高清的一个必要条件，运用视觉增强技术可以很好进行 HDR 色彩扩展，增强视频显示效果。

6. 风格迁移与颜色拓展

视觉增强还可以用来进行风格迁移，比如某些相机软件，可以将一些名画的风格迁移到用户所拍摄的照片上，实现照片的风格多样化。

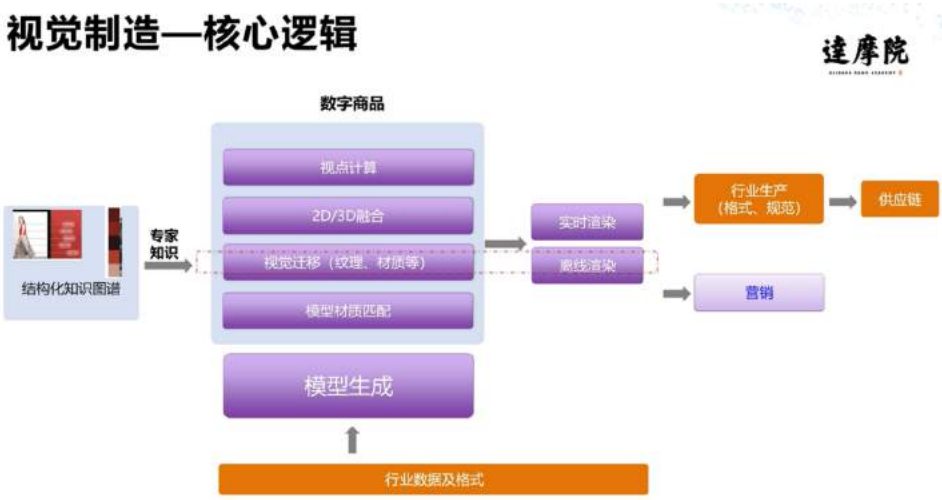
另外，视觉增强还可以进行颜色的拓展，比如下图所示的广告，可以同时产生不

同色彩搭配的广告，满足不同的需求和色彩的多样性。

六、视觉制造——由虚入实

前面所讲的基本上都是数字内容，那么能否将虚拟的和实体的关联起来呢？当然是可以的，比如下图所示的包装设计和服装设计两个例子，我们可以利用视觉制造技术来解决实际生产过程中面临的效率低、协同差、定制难等问题。

视觉制造的核心逻辑如下图所示。



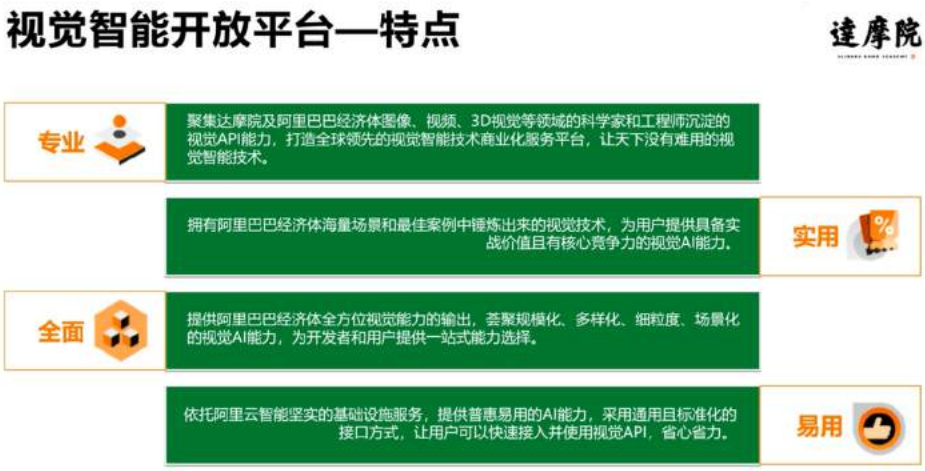
上面整个过程的用到了多种技术，比如包装几何生成、服装几何生成、材质纹理的多样化生成、视觉迁移及融合、多样性拓展等等。如下图所示，在得到物体或者商品的模型之后，利用 2D3D 融合，还可以将其与背景或者其他商品结合在一起，直接渲染效果图和商品的打样工作。同时，我们还可以完成从 3D 到 2D 的转换，形成一个闭环，对行业效率有了大大的提升。

七、视觉智能开放平台——万剑归宗

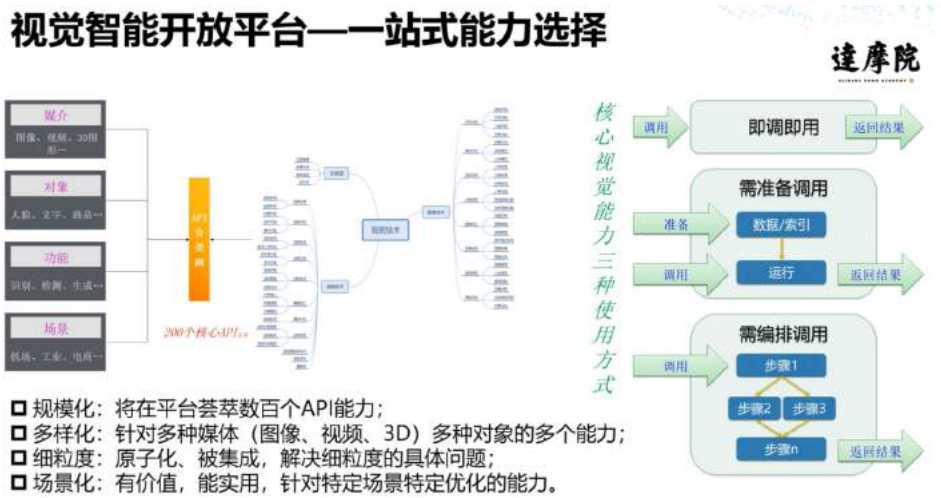
上面所提到的技术都可以在阿里巴巴的视觉智能开放平台（vision.aliyun.com）上找到，感兴趣的可以去尝试一下。

目前，该平台已经开放了 2 个多月，主要包含了如下图所示的多种能力，包括图像的和视频的，有 100 多种细分能力，实现了场景的全面覆盖。

该平台拥有专业、实用、全面、易用 4 大特点，并且可以进行一站式能力选择。



d



除此之外，视觉智能开放平台在公共云和专有云上都提供了多项服务，有着强大的供应链平台和基础设施，可以为用户提供省心省力的普惠服务。

如下图所示，针对一些场景，视觉智能开放平台提供了完整的场景解决方案，比如公共场所口罩佩戴检测系统（神荼）、视频自动广告植入系统、服饰趋势分析与辅助设计等等。

解决方案：视频自动广告植入系统

全自动完成视频内容的分析并植入指定广告内容，实现广告批量投放

◆解决思路：结合广告位检测、识别跟踪、视频分割、植入及渲染能力，打造全自动的视频广告检测与植入系统。
◆预期目标：实现视频内容的全自动广告分析与植入，使内容叠加增值，产生额外的点播营收。
◆应用技术：
I. 广告位检测：基于视觉理解与识别技术，判断视频中可以植入广告的位置，并完成定位与跟踪；
II. 广告场景识别：根据检测到的广告位，判断周围的场景、人物与事件，打上相应的广告标签，定向投放相宜的广告素材；
III. 视频分割：在有前景运动物体遮挡的镜头，以视觉分割技术完成人、车等物体的分割，并将其覆盖在植入的位置前；
IV. 植入与渲染：在视频素材中完成图像、视频的内容植入，并自动调整透视、曝光等外界干扰信息，实现无感知的视觉替换。
◆价值分析：传统手工植入广告模式周期长、效率低、成本高、安全性差，不能复制到海量在线视频及短视频。采用自动广告植入可以实现批量化自动投放，且结合场景化广告植入，可实现千人前面的差异化曝光，最大化内容价值。

解决方案：服饰趋势分析与辅助设计

智能分析服装的搭配要素与时尚属性，为设计师设计提供关键信息输入

◆解决思路：结合图像分割、商品属性识别、图像搜索、特征提取与匹配、风格迁移等技术，实现服装的设计趋势分析与辅助设计。
◆预期目标：获取服饰的特征属性，进行趋势分析，为设计师提供同款搜索、搭配、拓展等设计辅助工具。
◆应用技术：
I. 图像分割：从行业数据中分割模特或单品；
II. 商品属性识别：根据商品特征完成属性信息的识别获取，对商品打标；
III. 图像搜索：通过输入的服饰在素材库中进行特征匹配，寻找相同或相似款式；
IV. 风格迁移：对服饰的面料、印花进行色彩、纹理的迁移变化。
◆价值分析：当下服饰设计相关数据快速增长，以传统的人工处理、理解手段难以分析大量数据并掌握时尚趋势。通过视觉AI技术可以提取海量服饰数据的特征信息，并通过分析帮助设计师丰丰把握时尚趋势，从而通过设计手段实现爆品的打造。

更多的方法和实例大家可以到视觉智能开放平台官网查看和详细了解。

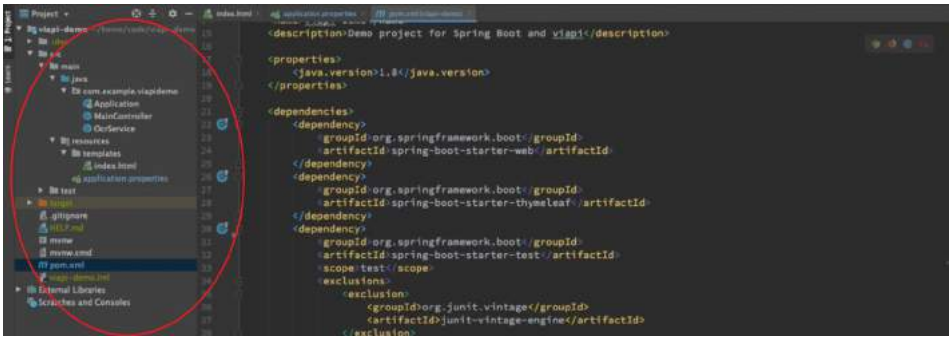
身份证识别系统搭建

在进行实践之前，我们先看一下最终的效果，如下图所示，该系统是一个简单的身份证识别系统。

用户上传身份证人像面和国徽面之后，点击开始识别，就可以得到身份证正反面的内容了。

一、项目简介

下图是这个项目的文件结构，它是通过 spring-boot-starter 创建的一个 spring-boot 项目。



项目中主要文件的介绍如下：

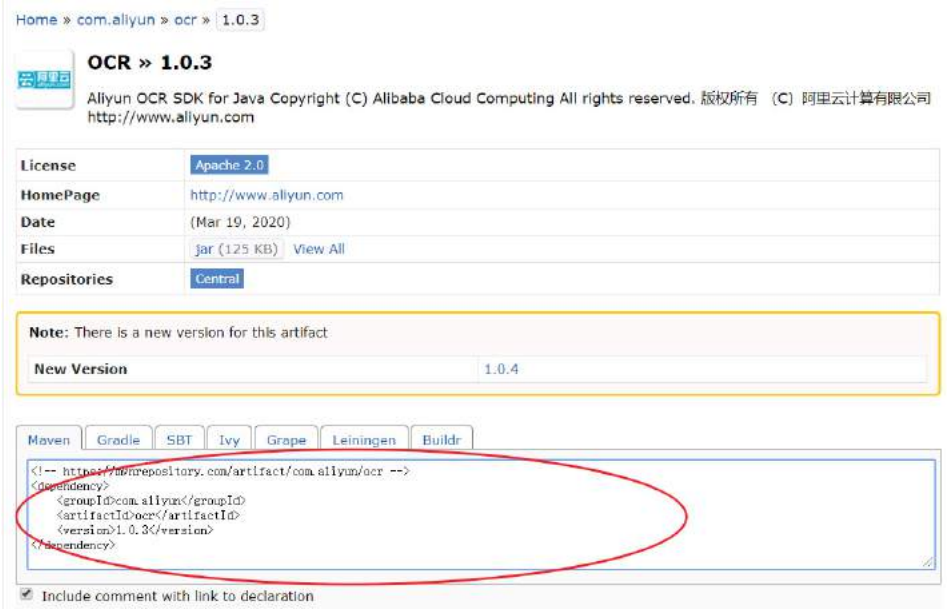
- Application：Spring-Boot 的一个启动类；
- MainController：控制器层，负责模版的渲染、路由等功能；
- OcrService：负责通过 SDK 调用视觉智能开放平台的 OCR 能力；
- index.html：基于 thymeleaf 的前端模版；
- application.properties：包含若干配置项的配置文件；
- pom.xml：pom 依赖。

二、如何获取视觉智能开放平台提供的 SDK？

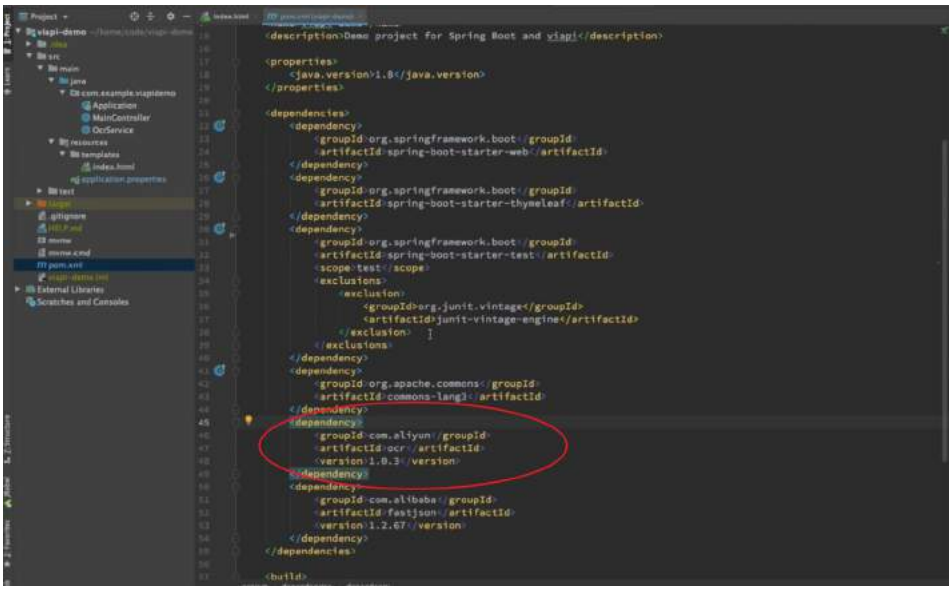
进入到视觉智能开放平台的官网 https://vision.aliyun.com/ 后往下拉，我们可以发现平台已经开放了包括人脸识别、文字识别、商品理解等在内的多项视觉 AI 能力，点击文字识别菜单，可以找到身份证识别项，然后我们点击进入，之后点击文档链接查看具体文档。

在文档页面，我们点击 SDK 参考，可以看到有两个 Java 的 SDK 说明，两个 SDK 的区别主要是新的 JavaSDK 支持本地上传图片，也就是说通过这个新的 SDK 可以直接把本地的图片交给视觉智能开放平台来进行 OCR 识别、人脸识别等任务。

点击 Java（支持本地上传）进入相应的说明页，我们可以看出其中有很多 SDK，我们需要找到需要的 OCR SDK。如下图所示，我们可以通过 https://mvnrepository.com/artifact/com.aliyun/ocr 来找到我们所需要的相应版本的 SDK，然后获取我们需要的 Maven 坐标。



在获取了 Maven 坐标之后，如下图所示，我们可以通过在 pom.xml 文件中添加 Maven 依赖安装 java SDK，这样便成功获取到了视觉智能开放平台提供的 SDK。

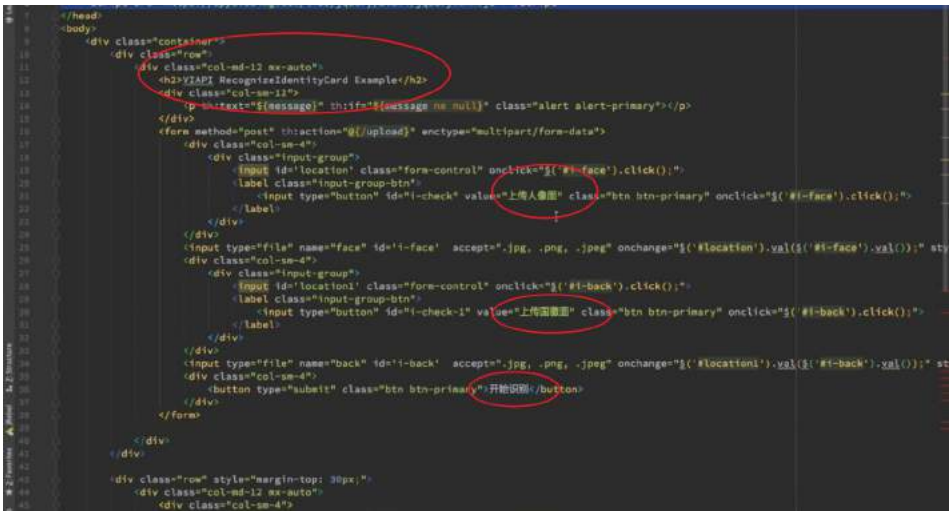


三、项目实现逻辑

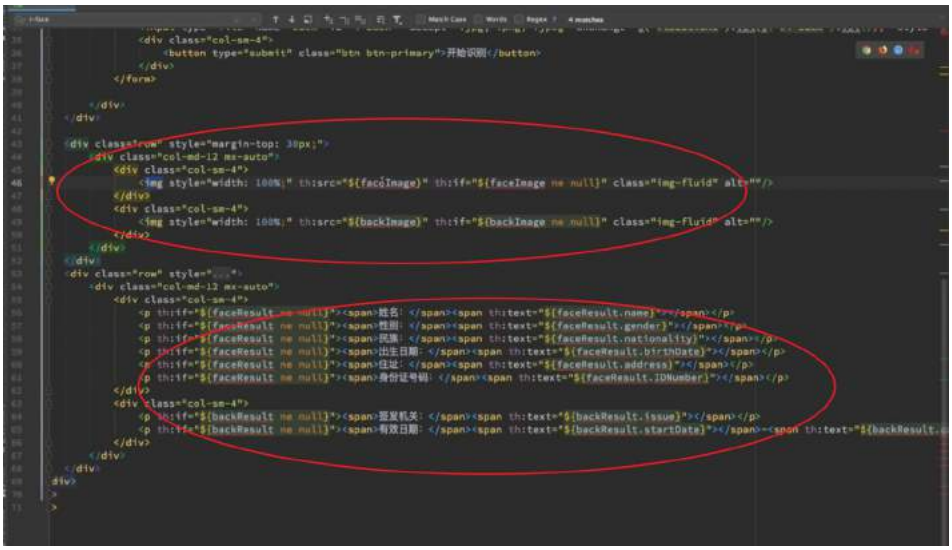
（一）前端

前端的实现是基于 thymeleaf 做的一个模版，其页面如下图所示，包括一个标题、2 个表单、2 个上传文件的组件以及一个开始识别的按钮。

下图所示是相应的标题、表单和按钮的实现代码。这里我们用到了 bootstrap 和 jquery 来进行页面的美化，我们用代理的方式来实现美化，相当于我们点击“上传人像面”的组件时候，会把表达上传到 input 组件中，再把 input 中的内容传给 form 表单，“上传国徽面”也是如此。在 input 组件中，我们限制了可以上传图片的类型，最前面还设置了一个 alert，在用户上传的图片出现问题的时候会进行提示或者报错。



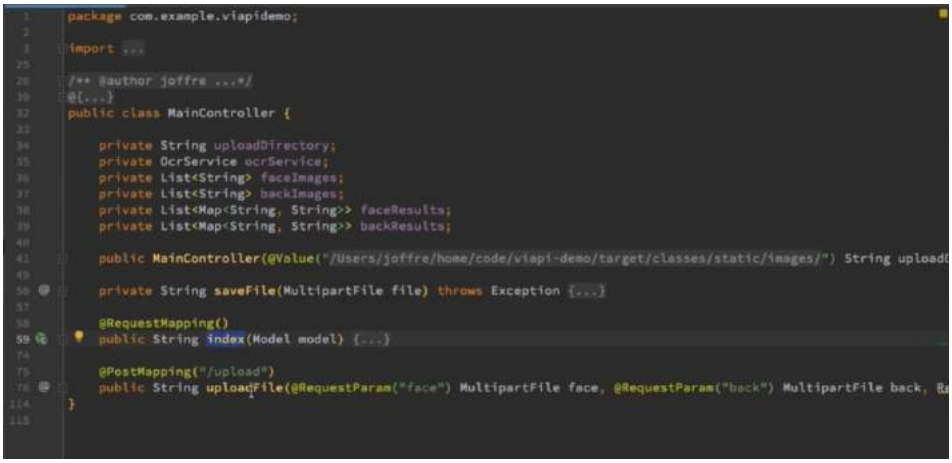
下图是在用户上传正确的图片并提交之后的相关代码逻辑，主要分为两部分：一是对上传的图片进行展示，二是对 OCR 识别出的信息进行展示，包括姓名、性别、民族、出生日期等信息。



（二）控制层

控制层主要包括如下几部分：

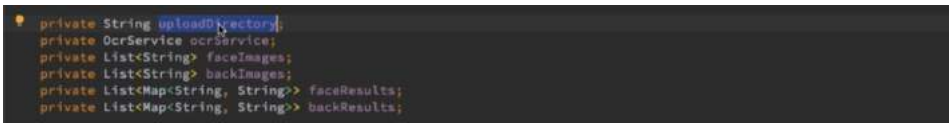
- 定义的若干个私有成员；
- MainCntroller；
- saveFile；
- index；
- uploadFile。



```
package com.example.viapidemo;

import ...

/** @author joffre ...*/
@{...}
public class MainController {

    private String uploadDirectory;
    private OcrService ocrService;
    private List<String> faceImages;
    private List<String> backImages;
    private List<Map<String, String>> faceResults;
    private List<Map<String, String>> backResults;

    public MainController(@Value("/Users/joffre/home/code/viapi-demo/target/classes/static/images/") String uploadD

    private String saveFile(MultipartFile file) throws Exception {...}

    @RequestMapping()
    public String index(Model model) {...}

    @PostMapping("/upload")
    public String uploadFile(@RequestParam("face") MultipartFile face, @RequestParam("back") MultipartFile back, Ra
}
```

（1）私有成员

私有成员变量如下图所示，包括上传图片文件本地保存地址（uploadDirectory）、调用视觉智能开放平台能力的封装（ocrService）、上传图片的缓存路径地址（faceImage、backImage）、识别结果的缓存（faceResults、backResults）。

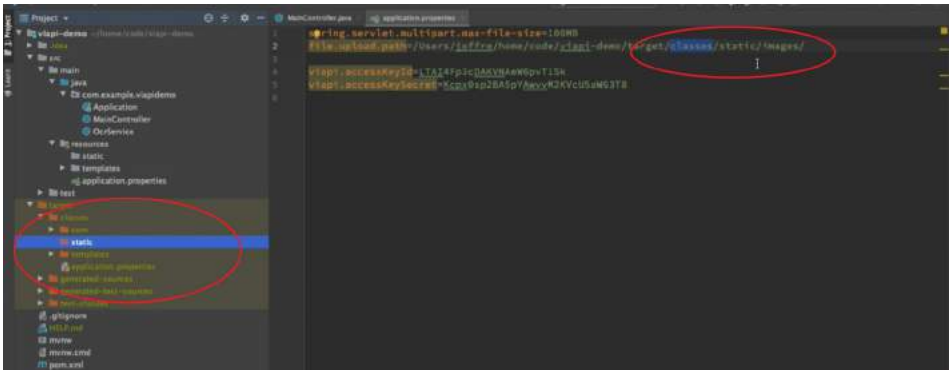


```
private String uploadDirectory;
private OcrService ocrService;
private List<String> faceImages;
private List<String> backImages;
private List<Map<String, String>> faceResults;
private List<Map<String, String>> backResults;
```

（2）MainController

MainController 构造方法的代码如下图所示，其中 file.upload.path 变量是在配置文件中的，是本机的一个上传文件的目录地址，大家可以在实践的时候根据自己的情况进行设置。这里有个小技巧，这里配置的目录地址其实是项目 target 目录下的一个子目录，这是因为 springboot 会默认取 target 下的 static 目录作为静态文件的地址，如果我们这么设置的话，上传和读取文件对应的是同一目录，可以更方便的进行文件的读取和存储操作。

```
public MainController(@Value("${file.upload.path}") String uploadDirectory, OcrService ocrService) {
    this.uploadDirectory = uploadDirectory;
    this.ocrService = ocrService;
    faceImages = new ArrayList<>();
    backImages = new ArrayList<>();
    faceResults = new ArrayList<>();
    backResults = new ArrayList<>();
}
```



（3）saveFile

saveFile 是一个私有方法，用来保存文件，其代码如下图所示。

```
private String saveFile(MultipartFile file) throws Exception {
    String suffix = StringUtils.substringAfterLast(file.getOriginalFilename(), separator: ".");
    String filename = UUID.randomUUID().toString() + "." + suffix;
    Path path = Paths.get( first: uploadDirectory + filename);
    Files.copy(file.getInputStream(), path, StandardCopyOption.REPLACE_EXISTING);
    return filename;
}
```

（4）index

index 是一个控制方法，是一开始进入页面时候用来加载模版的。首先是判断 faceImage 和 backImage 缓存是否一样，如果不一样，就全部清除掉；其次就是将已经上传的结果进行展示，起到一个重新刷新页面也不会丢失识别结果的作用。

```
@RequestMapping()
public String index(Model model) {
    if (faceImages.size() != backImages.size()) {
        faceImages.clear();
        backImages.clear();
        faceResults.clear();
        backResults.clear();
    }
    if (!CollectionUtils.isEmpty(faceImages) && faceImages.size() == backImages.size()) {
        model.addAttribute( s: "faceImage", faceImages.get(faceImages.size() - 1));
        model.addAttribute( s: "faceResult", faceResults.get(faceResults.size() - 1));
        model.addAttribute( s: "backImage", backImages.get(backImages.size() - 1));
        model.addAttribute( s: "backResult", backResults.get(backResults.size() - 1));
    }
    return "index";
}
```

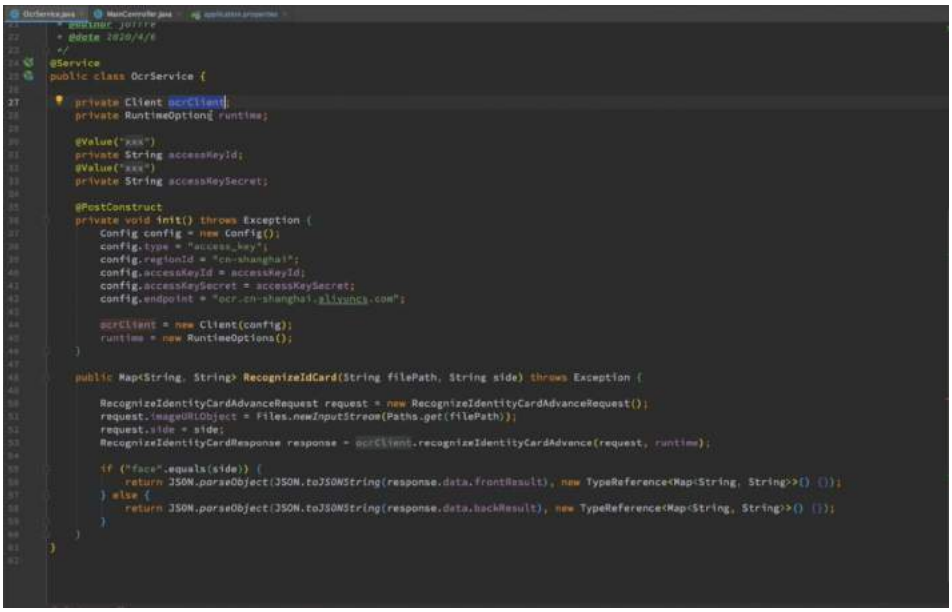
（5）uploadFile

uploadFile 也是一个控制方法，有三个参数，一个正面的人像面的文件，一个背面的国徽面的文件，还有一个用来重定向的参数。首先，判断两张图片是否是空的，如果是空的，将会重定向到 index，相当于重新进入首页，并且会有一个 message 来提示用户必须要上传一个文件，否则是无效的；接下来判断上传目录是否存在，如果不存在就递归的进行创建；然后如果人像面的文件不为空，将其保存到本地，紧接着调用视觉智能开放平台的能力去进行识别，在拿到结果之后将上传的图片和识别结果加入到缓存池中，国徽面的处理方式也类似；最后几行代码是异常的处理，并且会告知用户是哪种报错。

```
@PostMapping("/upload")
public String uploadFile(@RequestParam("face") MultipartFile face, @RequestParam("back") MultipartFile back, RedirectAttributes attributes)
    if (face.isEmpty() || back.isEmpty()) {
        attributes.addFlashAttribute( attributeName: "message",  attributeValue: "Please select a file to upload.");
        return "redirect:/";
    }

    String errorMessage = null;
    try {
        Path dir = Paths.get(uploadDirectory);
        if (!Files.exists(dir)) {
            Files.createDirectories(dir);
        }

        if (!face.isEmpty()) {
            String filename = saveFile(face);
            Map<String, String> res = ocrService.RecognizeIdCard( filePath: uploadDirectory + filename,  side: "face");
            faceImages.add("/images/" + filename);
            faceResults.add(res);
        }
        if (!back.isEmpty()) {
            String filename = saveFile(back);
            Map<String, String> res = ocrService.RecognizeIdCard( filePath: uploadDirectory + filename,  side: "back");
            backImages.add("/images/" + filename);
            backResults.add(res);
        }
    } catch (TeaException e) {
        e.printStackTrace();
        errorMessage = JSON.toJSONString(e.getData());
    } catch (Exception e) {
        e.printStackTrace();
        errorMessage = e.getMessage();
    }

    if (StringUtils.isNotBlank(errorMessage)) {
        attributes.addFlashAttribute( attributeName: "message", errorMessage);
    }
    return "redirect:/";
}
}
```

（三）OcrService

CcrService 的实现代码如下所示。首先，是 4 个私有变量，ocrClient、runtime、accessKeyId 和 accessKeySecret，其中 accessKeyId 和 accessKeySecret 要在配置文件中进行配置；其次是一个初始化方法，初始化 Client 和 runtime；接下来是真正调用视觉智能开放平台身份证识别的方法，具体来说，在设置好 request 之后，我们将 request 和 runtime 作为参数传递给 ocrClient 来调用身份证识别的方法，之后我们得到相应的 response，也就是识别的结果，之后我们将结果处理成哈希 Map 返回给控制层。



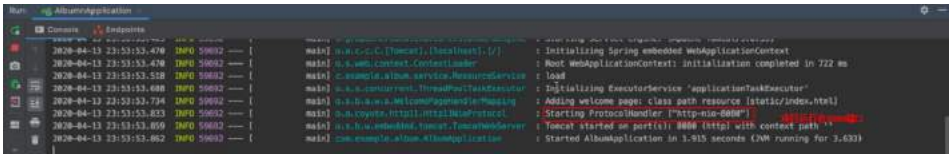
```java
 * @author jorrie
 * @date 2020/4/6
 */
@Service
public class OcrService {

    private Client ocrClient;
    private RuntimeOptions runtime;

    @Value("xxx")
    private String accessKeyId;
    @Value("xxx")
    private String accessKeySecret;

    @PostConstruct
    private void init() throws Exception {
        Config config = new Config();
        config.type = "access_key";
        config.regionId = "cn-shanghai";
        config.accessKeyId = accessKeyId;
        config.accessKeySecret = accessKeySecret;
        config.endpoint = "ocr.cn-shanghai.aliyuncs.com";

        ocrClient = new Client(config);
        runtime = new RuntimeOptions();
    }

    public Map<String, String> RecognizeIdCard(String filePath, String side) throws Exception {

        RecognizeIdentityCardAdvanceRequest request = new RecognizeIdentityCardAdvanceRequest();
        request.imageURLObject = Files.newInputStream(Paths.get(filePath));
        request.side = side;
        RecognizeIdentityCardResponse response = ocrClient.recognizeIdentityCardAdvance(request, runtime);

        if ("face".equals(side)) {
            return JSON.parseObject(JSON.toJSONString(response.data.frontResult), new TypeReference<Map<String, String>>() {});
        } else {
            return JSON.parseObject(JSON.toJSONString(response.data.backResult), new TypeReference<Map<String, String>>() {});
        }
    }
}
```

控制层拿到返回的结果之后，结合前端的优化进行结果的展示，至此，我们便完成了一个身份证识别系统的构建。

电子相册搭建（人脸、表情识别）

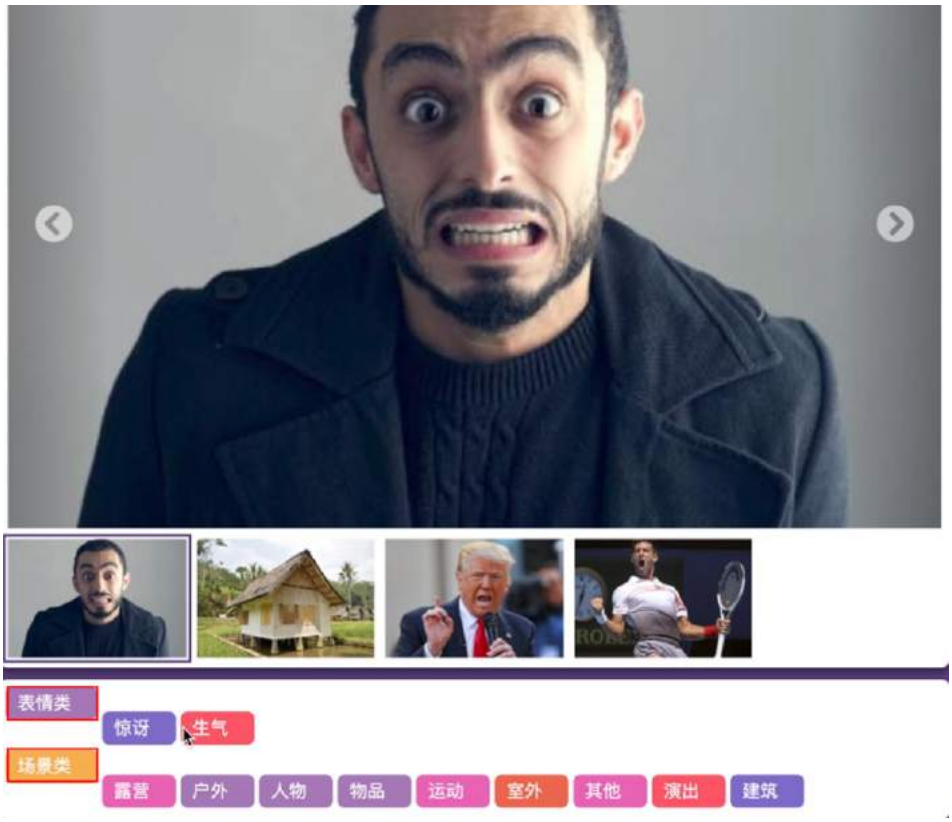
一、项目简介

我们首先来看一下项目运行出来的效果：访问程序运营端口：127.0.0.1.8080/index.html



所以我们看到是三块空白，然后这边的话是我们可以通过点击上传，也可以通过拖拽去上传图片。下面我们来看一下效果，

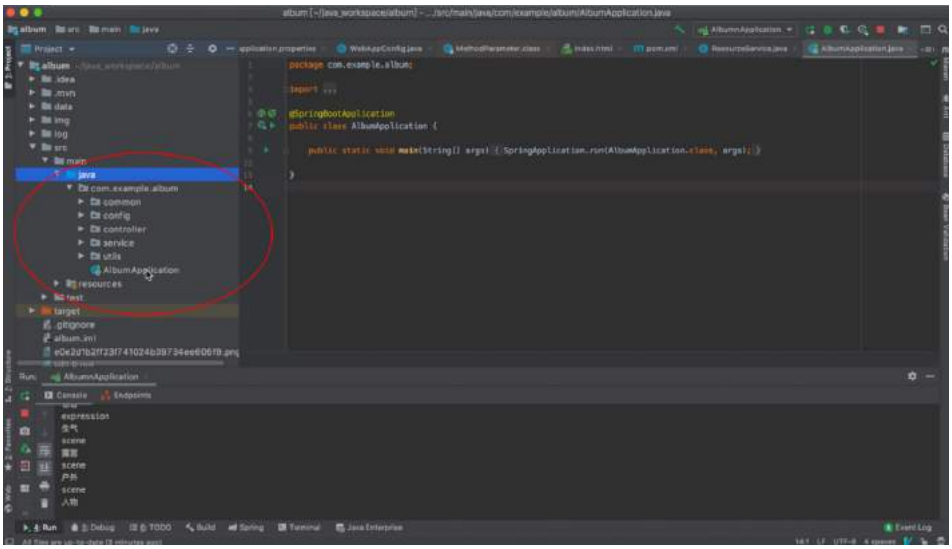
本次算法进行了表情识别和场景识别：



第一部分我们多张图片拖拽上传，第二部分实现一个轮播图，这时候可以看到有吃惊的表情，生气的表情以及露营户外或者是人物，可以看到对这个图片分类利用了两类的视觉的算法，一个是表情识别，然后另外一个是场景识别。然后我们可以去查看图片。

二、项目实现

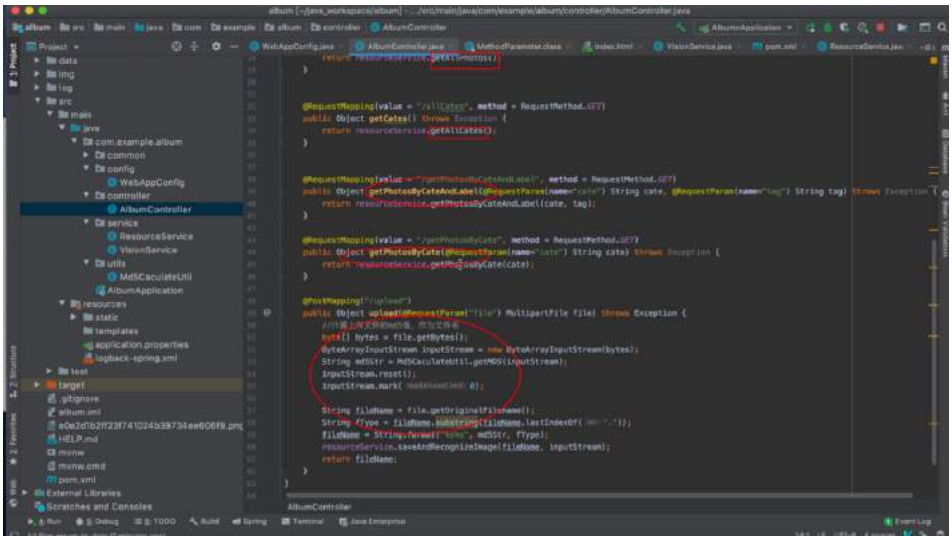
然后现在来看一下代码，先讲解一下它的整个结构：



- Application：启动的入口函数，
- common：一般就是存放公共的类或者常量，或者枚举值，
- config：装载或者是数据库的配置，我们都会放在 config 下面，webAP-Pconfig 是对我们的静态资源，比如说 css、js 还有一些图片，做了一个映射，比如说 static，然后我们把它映射到 classpath:/static 目录下。
- Controller：我们接收外部的请求，比如参数校验，之后通过接口调用或得到的数据返回给前端
- Service: 把不同的请求 , 不同的服务把它抽象成一个 service，本项目有两个 service，也可以认为是两个模块。
- Utils: 存放公共的类或者工具函数。

接下来看一下具体的实现：

先从 Controller 来，看一下实现了哪些功能。

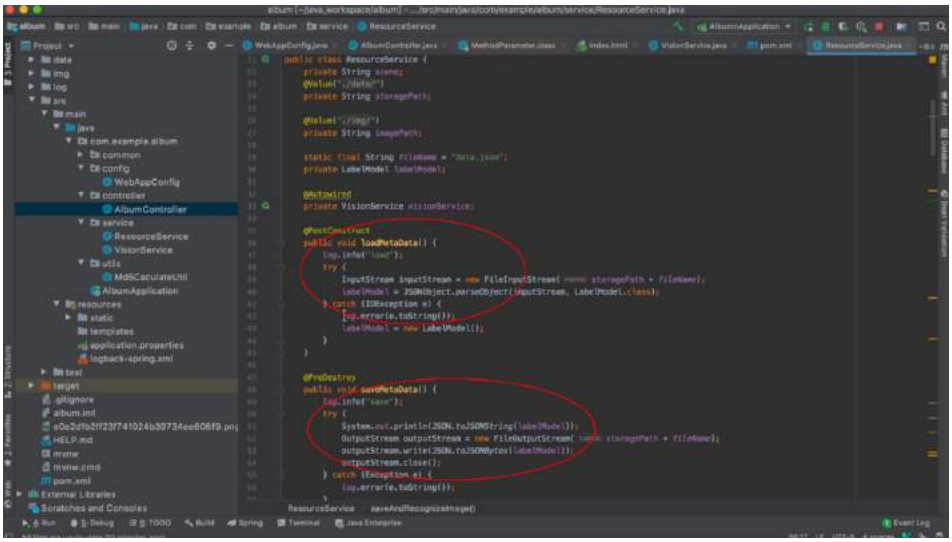


可以获取图片、获取分类，通过分类以及标签获取图片，通过分类获取图片列表以及上传文件。

在上传文件函数需要完成一个上传，把上传的文件保存到文件当中，需要调用两个识别的函数，表情识别，场景识别。

另外需要注意一下，通过 multipartFile 获取的是一个文件的输入流，输入流的话它只能读取一次，然后如果要重复读的话就会是空。 所以我们把它转化成一个 ByteArrayInputStream 流，然后我们在每次用完之后，我们可以把它进行一个 reset()，之后可以把流保存在我们的内存当中，一般不推荐使用这种方式，因为如果图片的比较大的话，可能会占用太多的内存空间。此外由于我们上传的图片可能会有重名，为了避免重名，我们对这个图片的 input 的流，我们给它求一个 md5 值作为文件名。

接着来看一下 ResourceService:



ResourceService 是一个资源的管理器，先来看两个函数，Postconstruct 注解，我们会先去执行这个函数，取出保存在本地的数组并且加载到内存中去。PreDestroy 注解，在销毁对象的时候，把这些数据保存到本地文件，也就是说保存在这个 data 上面，有一个 data.json 数据，来看一下整个的数据存储的结构。

allImg 将所有上传的图片，放在数组里面。

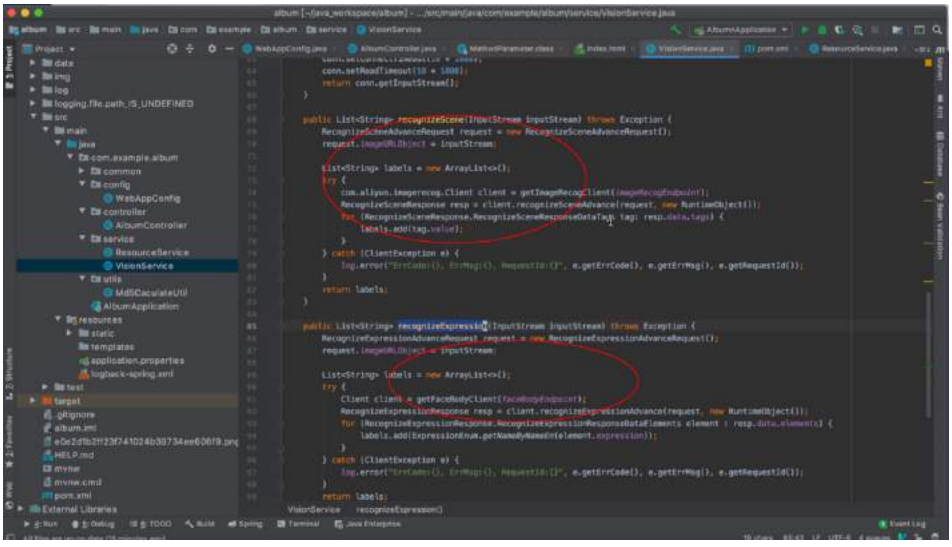
cateMap 之后又分成两个场景 expression、scene，可以看到不同的分类，表

情识别，场景识别，里面存放所有识别出来的表情，比如说惊讶、生气、开心，场景识别，存放运动、户外，这样存储是为了我们能够快速的进行查询，比如说可以通过分类，快速的找到某一个分类下面的所有的识别，这个是 map 的作用。

expressionMap 是表示某类标签下面都包括哪些图片，senceMap 也是同样的意思。

imgLabels 是一个反向的查找的过程，即通过一个图片，识别出来了哪些场景，比如说我们可以看到它可能这张图片它可能直接识别出来了，它是属于人物场景，属于运动场景，然后属于生气的表情，属于演出等场景。

接着来看一下 VisionService。



VisionService，识别场景，表情。我们上传图片流，我们把整个图片流传到接口里面，然后从服务端去识别这个场景。

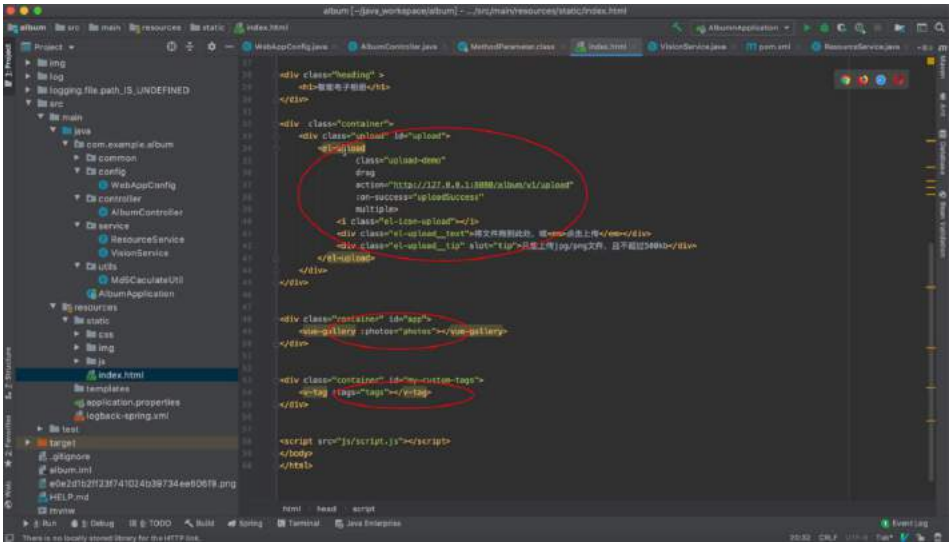
我们是有两种方式，早期的版本我们是通过 url 这种方式去上传的，但是有一个限制就是说我们必须是利用这种 oss 的对象才能够识别，新的版本的 SDK 我们就是开始支持通过本地上传图片来进行识别。

请求参数

名称	类型	是否必选	示例值	描述
Action	String	是	RecognizeExpression	系统规定参数。取值：RecognizeExpression。
ImageURL	String	是	https://viapi-test.oss-cn-shanghai.aliyuncs.com/test/facebody/RecognizeBrow/brow.jpg	图片URL地址。

以上这个是后端的这些功能。下面来看一下前端页面的实现：

我们前端页面是通过 vue，然后加上 element-ui 这个两个组件来实现我们的前端的逻辑。我们大概可以看一下整个的结构，它可以分成三部分。

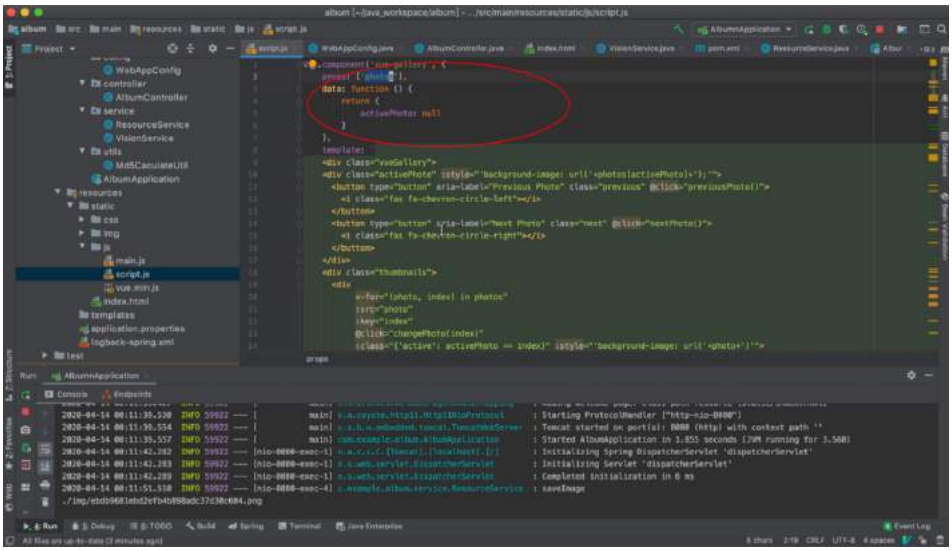


一部分就是我们通过一个上传的组建来去实现我们刚才拖拽上传，以及点选。

找到一个图片，我们可以看到两种方式，一种拖拽，一种是点击上传，这是通过一个组件来实现的。我们大概来介绍组件的实现，这是有不同的功能区分的。action 是在上传图片要去访问后端的 upload 接口，之后有一个事件，是在成功之后应该做哪些事情，也就是对应的要执行的函数。 比如 UploadSuccess 指上传成功之后，我们要刷新页面，添加下面的标签分类等。

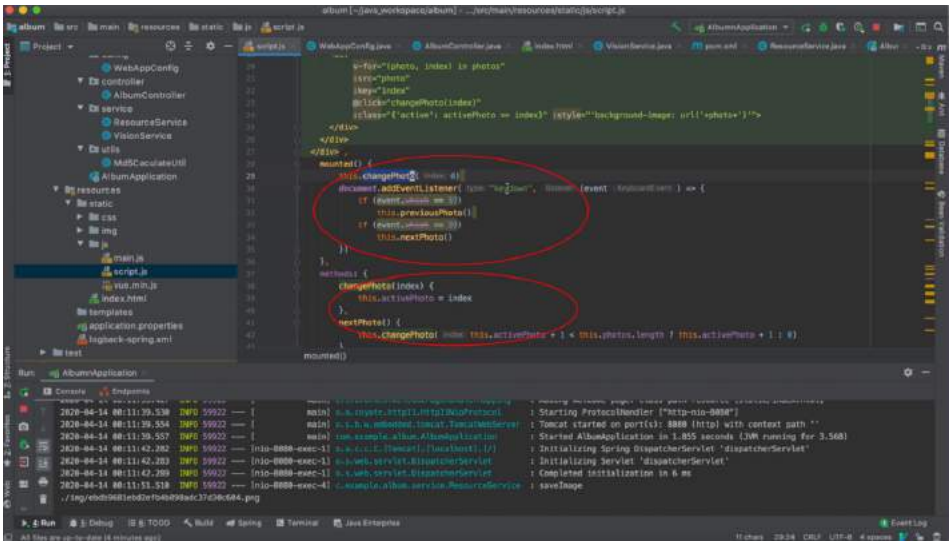
第二部分就是实现轮播图的部分：

然后以及下面列表，通过是 vue 的一个组件。



vue-gallery, 实现自定义的标签名，之后定义了一个属性：photos。photos 从后台去取回数据，之后把它渲染到前端的 html 页面当中。由于我们这是一个组件，它会有对应的自己一个模板，这个就是我们整个的模板部分，之后对它进行背景图片的处理，然后以及一些事件的定义。

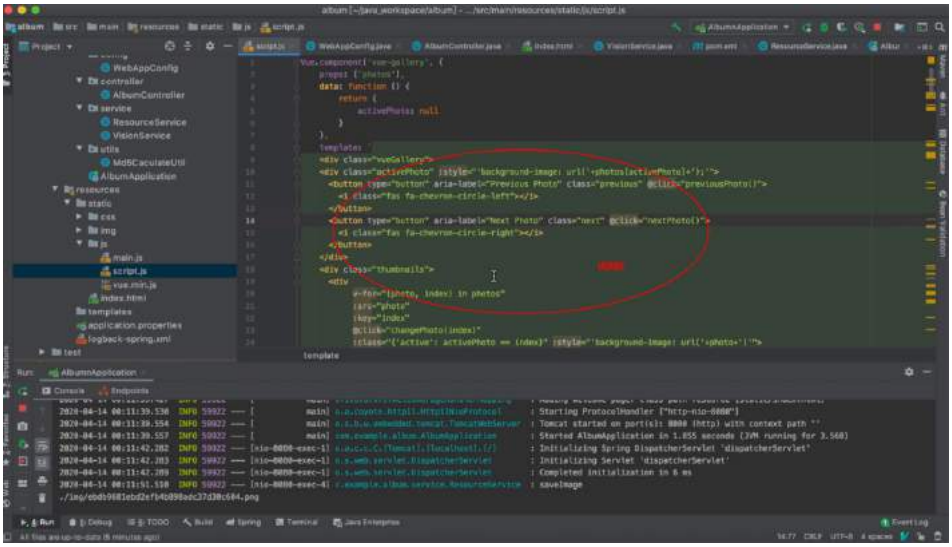
vue，它的整个是一个事件的数据流，通过数据的不同的变化，然后我们就可以去触发它的渲染，比如上传一张图片，它是可以对应不同的组件进行交互，利用不同的标签，事件来驱动数据的变化。

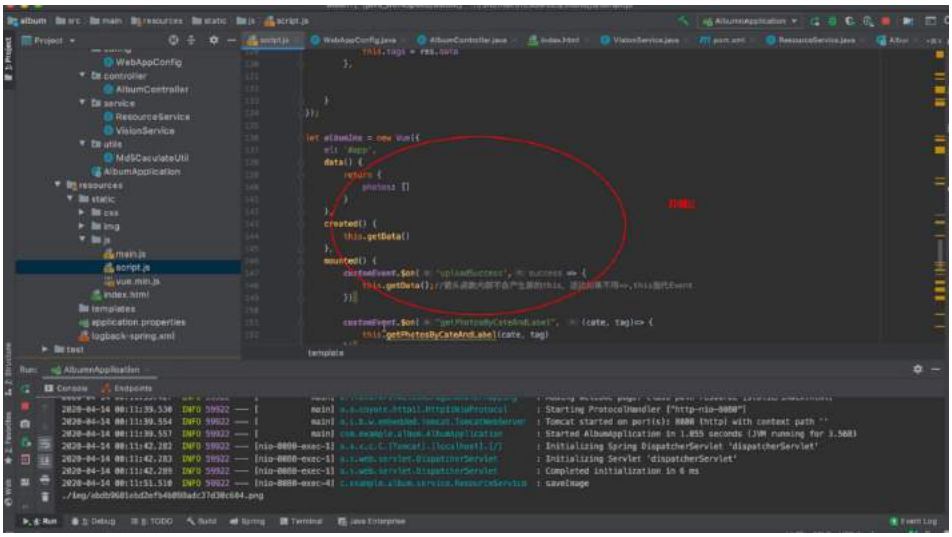


mounted 是会定义一些事件，比如图片变化，怎么调用，以及我们会监听按下不同的键，应该做哪些操作？然后是前张图片还是后一张图片等。

methods 定义的就是一些方法，比如点击一个 next photo，我们就会去访问下一张图片。这些变化是对应的。

vue 有自己的模板语言。

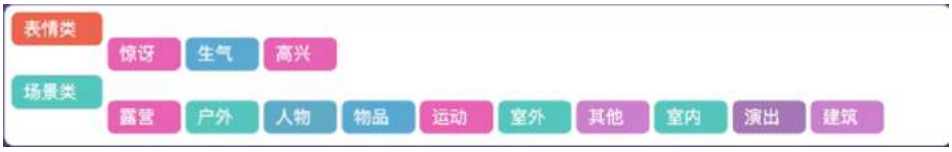




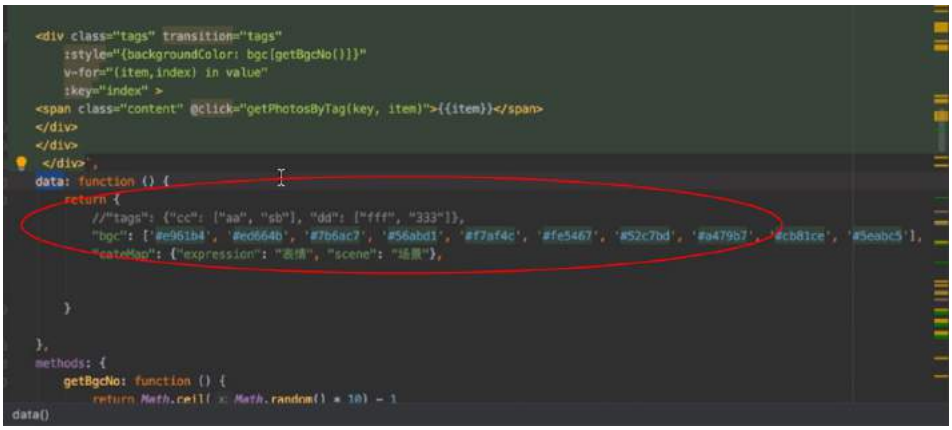
```
<div class="container" id="app">
    <vue-gallery :photos="photos"></vue-gallery>
</div>

<div class="container" id="my-custom-tags">
    <v-tag :tags="tags"></v-tag>
```

el 是一个 ID 的绑定，比如 #app, 在这个标签上层父级，定义了一个 ID 叫 app，这时候可以把组件放在父级 div 下面，然后在这个里面去根据模板进行渲染。

第三部分是自定义标签的组件。



V-tag 对应的是这块，我们把它分成不同的分类，下面有不同的标签是组件来实现的。



Data 是指在初始化的时候，需要进行哪些渲染。这里面自定义了不同的颜色。cateMap，将返回的英文转化成汉字去显示出来。

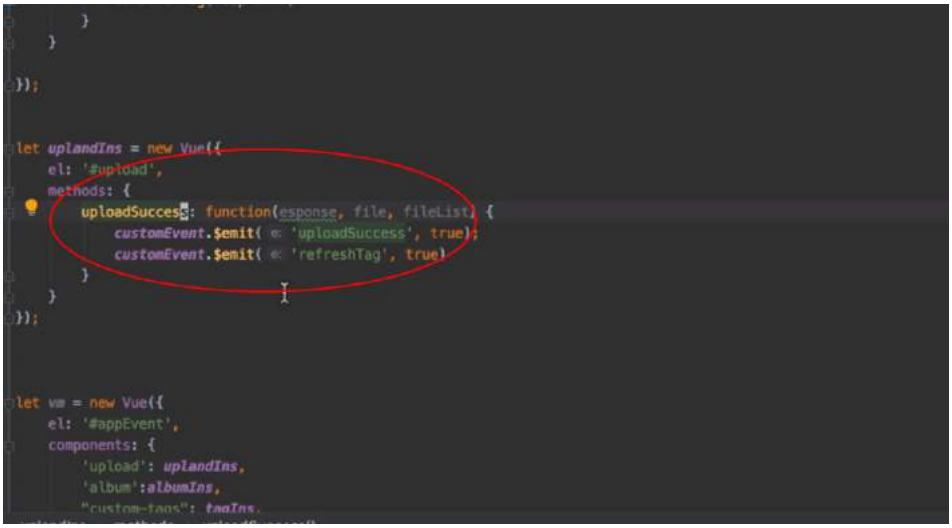
它实现了几个方法：

去取一个不同的 ID，然后给它选成不同的颜色，通过取一个随机的下标，然后去把它选成不同的颜色。

Tag 是一个两层 map 结构。就是第一层这里面某比每层结构，表情，下面有几种表情，场景下面有几种场景，这是一个两层的 map 结构。

所以在渲染的时候，会对两层的 map 进行渲染，第一层完成之后，再渲染第二层数据。

实现不同的触发，上传文件，成功之后要进行不同的事件操作：



因为要实现不同的组件之间进行通信，自定义了一个虚拟的 Event 的 vue 的实例，然后通过 Event 将所有的不同的 vue 实例串联起来。比如 uploadSuccess 去发送一个事件。通过 upload vue 实例去给其他的实例去发事件，在上传成功之后，加载到轮播图里面。

on 就对应去接收 emit 传过来的事件。如果我们得到了 uploadSuccess 事件，那么就会通过从后端去获取数据，对这个页面进行一个刷新操作。

refresh 也是同样的道理，上传成功之后，要对整个的 tag 进行一个刷新，把新识别出来的不同的场景，不同标签，进行一个刷新。

前端的实现逻辑就是这样的。

阿里云 开发者社区

阿里云开发者“藏经阁”
海量免费电子书下载

扫码加入高校计划