

1.创建小程序项目·开通云开发服务

1.1 创建QuickStart小程序项目

确保自己的开发者工具是最新的，安装完开发者工具之后，我们使用微信扫码登录开发者工具，然后使用开发者工具新建一个小程序的项目：



- **项目名称**：这个可以根据自己的需要任意填写；**目录**：大家可以先在电脑上新建一个空文件夹，然后选择它；
- **AppID**：就是之前我们找到的AppID(小程序ID)（也可以下拉选择AppID）
- **开发模式** 为小程序（默认），
- **后端服务** 选择小程序·云开发
- 点击 **新建** 确认之后就能在开发者工具的模拟器里看到**云开发QuickStart小程序**，在编辑器里看到这个小程序的源代码。

1.2 开通云开发服务

点击微信开发者工具的“云开发”图标，在弹出框里点击“开通”，同意协议后，会弹出**创建环境**的对话框。这时会要求你输入**环境名称**和**环境ID**，以及当前云开发的基础环境配额（基础配额免费，而且足够你使用哦）。

按照对话框提示的要求填写完之后，点击创建，会初始化环境，环境初始化成功后会自动弹出云开发控制台，这样我们的云开发服务就开通啦。大家可以花**两分钟左右**的时间熟悉一下云开发控制台的界面。



2. 配置项目的环境ID

2.1 找到云开发的环境ID

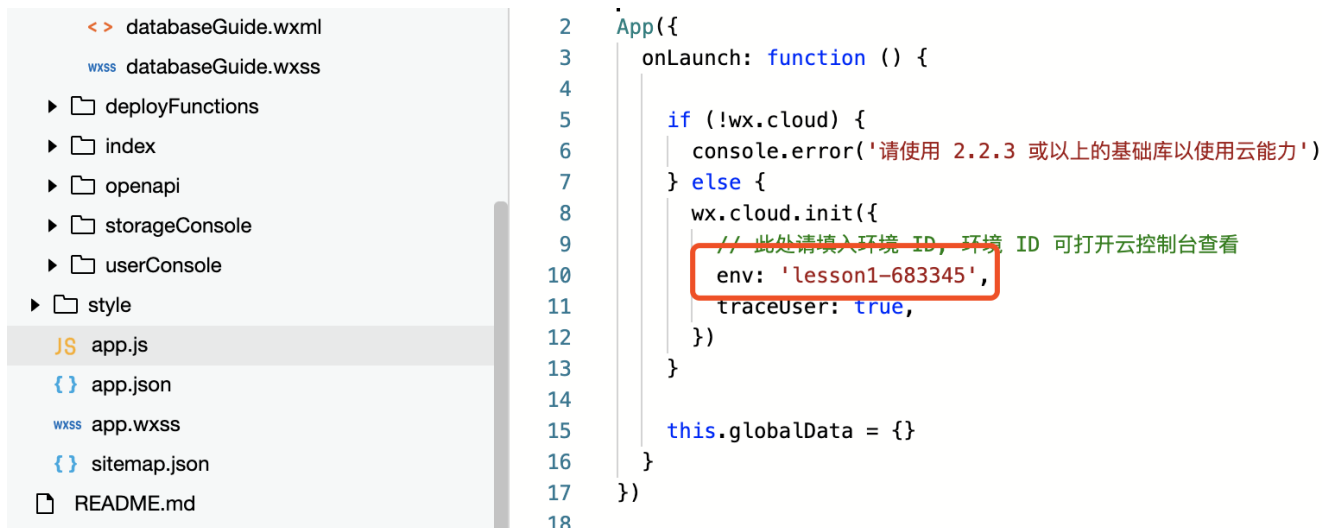
点击云开发控制台窗口里的**设置**图标，在**环境变量**的标签页找到**环境名称**和**环境ID**。



当云开发服务开通后，我们可以在小程序源代码cloudfunctions文件夹名看到你的环境

名称。如果在cloudfunctions文件夹名显示的不是环境名称，而是“**未指定环境**”，可以鼠标右键该文件夹，选择“**更多设置**”，然后再点击“**设置**”小图标，选择环境并确定。

2.2 指定小程序的云开发环境



在开发者工具中打开源代码文件夹**miniprogram**里的app.js，找到如下代码：

```
wx.cloud.init({
  // 此处请填入环境 ID, 环境 ID 可打开云控制台查看
  env: 'my-env-id',
  traceUser: true,
})
```

在env: 'my-env-id'处改成你的环境ID。

3. 部署并上传云函数

3.1 下载云函数依赖环境NodeJS

NodeJS是在服务端运行JavaScript的运行环境，云开发所使用的服务端环境就是NodeJS。**npm**是Node包管理器，通过npm，我们可以非常方便的安装云开发所需要的依赖包。

下载地址：[Nodejs下载地址](#)

大家可以根据电脑的操作系统下载相应的NodeJS安装包并安装（**安装时不要修改安装目录，啥也别管直接next安装即可**）。打开电脑终端（Windows电脑为**cmd**命令提示符，Mac电脑为**终端Terminal**），然后逐行输入并按Enter执行下面的代码：

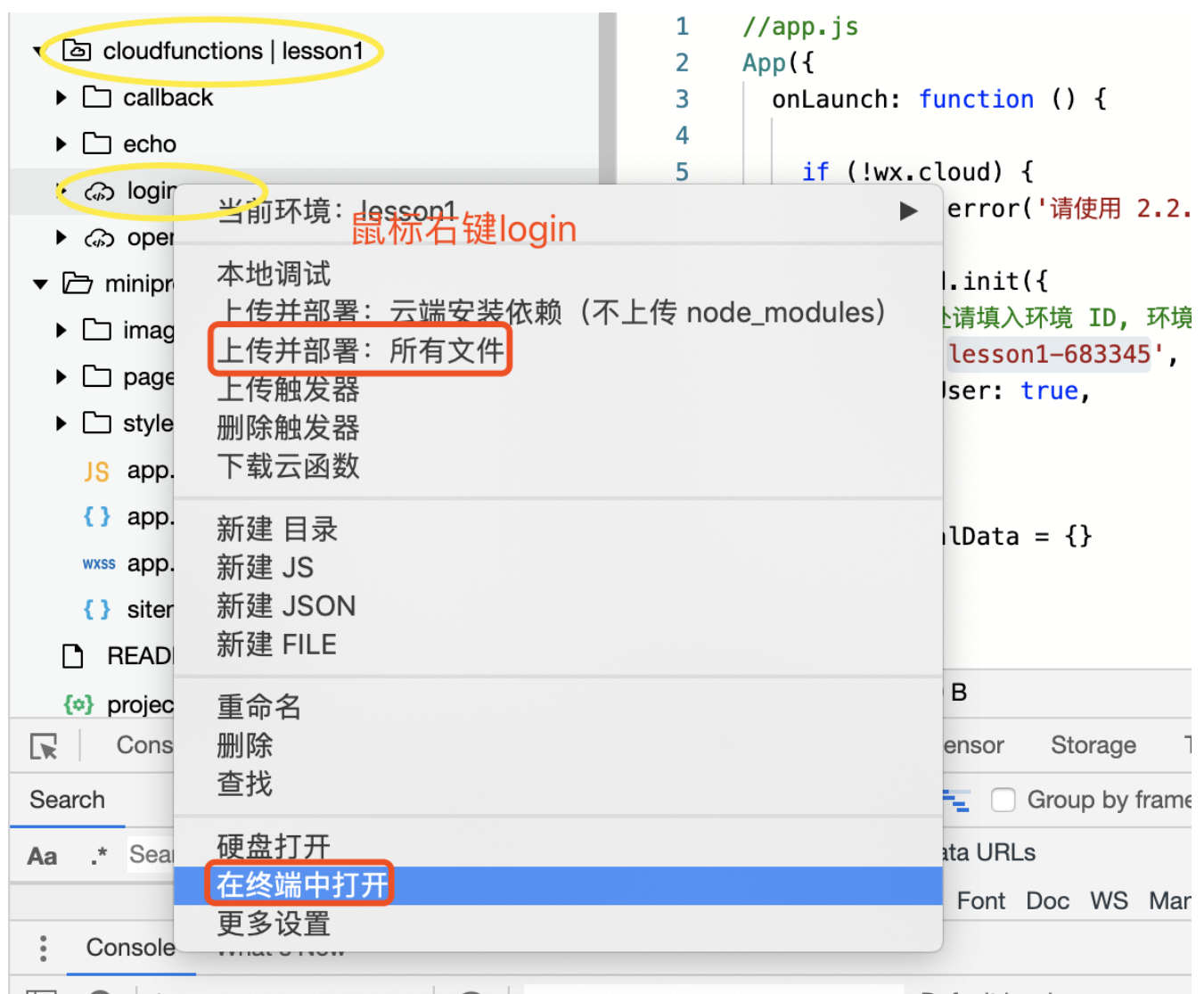
```
node --version
npm --version
```

```
[(base) lidongdeMacBook-Pro:~ lidong$ node --version
v10.15.0
[(base) lidongdeMacBook-Pro:~ lidong$ npm --version
6.10.1
```

如果显示为v10.15.0以及6.10.0（可能你的版本号会有所不同），表示你的Nodejs环境已经安装成功。

3.2 部署并上传云函数

cloudfunctions文件夹图标里有朵小云，表示这就是**云函数根目录**。展开cloudfunctions，我们可以看到里面有login、openapi等文件夹，这些就是**云函数目录**。



使用鼠标右键其中的一个云函数目录比如login，在右键菜单中选择**在终端中打开**，打开后在终端中输入以下代码并按Enter回车执行：

```
npm install
```

如果显示“npm不是内部或外部命令”，你需要关闭微信开发者工具启动的终端，而是重新打开一个终端窗口，并在里面输入`cd /D 你的云函数目录`进入云函数目录，比如`cd /D C:\download\tcb-project\cloudfunctions\login`进入login的云函数目录，然后再来执行`npm install`命令。

这时候会下载云函数的依赖模块，下载完成后，再右键login云函数目录，点击“**创建并部署：所有文件**”，这时会把本地的云函数上传到云端，上传成功后在login云函数目录图标会变成一朵小云。

在开发者工具的工具栏上点击“**云开发**”图标会打开**云开发控制台**，在云开发控制台点击**云函数**图标，就能在云函数列表里看到我们上传好的“login”云函数啦。

3.3 部署并上传其他云函数

接下来我们按照这样的流程把**其他所有云函数**（如openapi）都部署都上传，也就是：

- 右键云函数目录，选择在终端中打开，输入`npm install`命令下载依赖文件；
- 然后再右键云函数目录，点击“**创建并部署：所有文件**”

4. 下载并导入AI项目的源代码

4.1 下载AI项目并解压 点击下载此次活动的**AI项目源代码压缩包**，并解压，可以看到tcb-Demo-AICamera-master文件夹下有两个文件夹分别为init（**放置着项目的完整代码**）和intact（**此次活动的实战代码**）。

4.2 导入AI项目 点击开发者工具工具栏**项目-导入项目**，**项目名称**可以任意填写比如“AI人脸识别”，项目路径为你之前建好的AICamera文件夹下tcb-Demo-AICamera里面的**intact文件夹**

5. 配置AI项目

5.1 配置项目的环境ID

还记得怎么找你的环境ID么，打开client文件夹里的app.js，找到如下代码，填写一下你自己的环境ID。

```
wx.cloud.init({
  traceUser: true,
  envId: "learn-snoop"
});
```

5.2 添加pictures集合并设置权限

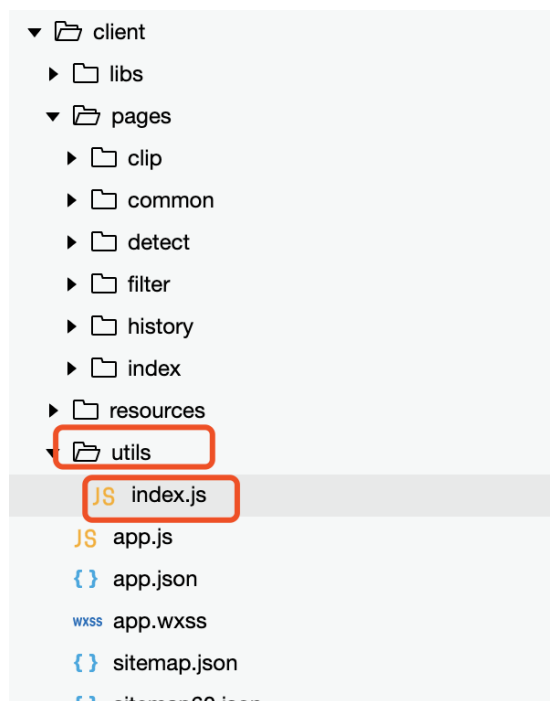


AI小程序会使用到数据库，打开云开发控制台，在数据库管理页新建一个pictures集合，然后将pictures集合的权限设置为所有用户可读，仅创建者可读写。

6. 上传图片到云存储功能

使用微信开发者工具打开client/utils/index.js将下面的代码复制粘贴到第11行处，添加了之后，上传人脸的按钮就可以使用啦，点击上传人脸，选择一张照片，就会把该

照片上传到云存储里。



```
1 export function uploadImage(filename) {
2   return new Promise((resolve, reject) => {
3     const dotPosition = fileName.lastIndexOf(".");
4     const extension = fileName.slice(dotPosition);
5     const cloudPath = `${Date.now()}-${Math.floor(
6       Math.random(0, 1) * 10000
7     )}${extension}`;
8
9     wx.cloud.uploadFile({
10      cloudPath,
11      filePath: fileName,
12      success: res => {
13        resolve(res);
14      },
15      fail: () => {
16        wx.hideLoading();
17        reject();
18      }
19    });
20  });
21 }
22 }
```

```
wx.cloud.uploadFile({
  cloudPath,
  filePath: fileName,
  success: res => {
    resolve(res);
  },
  fail: () => {
    wx.hideLoading();
    reject();
  }
});
```

这段代码主要是使用wx.cloud.uploadFile将本地资源上传到云存储空间。

7. 调用AI分析与人脸检测云函数

使用微信开发者工具打开pages/detect/index.js，将下面的代码复制粘贴到第100行处，也就是粘贴到callFunction()函数的try{}内：



```
let { result } = await wx.cloud.callFunction({  
  name: "tcbService-ai-detectFace",  
  data: {  
    FileID: this.data.fileID  
  }  
});  
wx.hideLoading();  
  
if (!result.code && result.data) {  
  this.setData(  
    {  
      faceRects: this.getFaceRects(result.data)  
    },  
    () => {  
      this.triggerEvent("finish", result.data);  
    }  
  );  
} else {  
  throw result;  
}
```

这段函数主要是调用tcbService-ai-detectFace云函数。

8. 照片的滤镜处理

8.1 怀旧风格滤镜

使用微信开发者工具打开pages/filter/index.js将下面代码粘贴到第111行处，也就是添加到handleOldTap函数内：



```
for (let i = 0; i < data.length; i += 4) {
  let r = originImageData.data[i];
  let g = originImageData.data[i + 1];
  let b = originImageData.data[i + 2];
  data[i] = 0.393 * r + 0.769 * g + 0.189 * b;
  data[i + 1] = 0.349 * r + 0.686 * g + 0.168 * b;
  data[i + 2] = 0.272 * r + 0.534 * g + 0.131 * b;
  data[i + 3] = originImageData.data[i + 3];
}
```

8.2 图片的毛玻璃滤镜

使用开发者工具打开 `pages/filter/util.js` 将下面代码粘贴到第52行处，也就是 `SmoothY` 函数内：



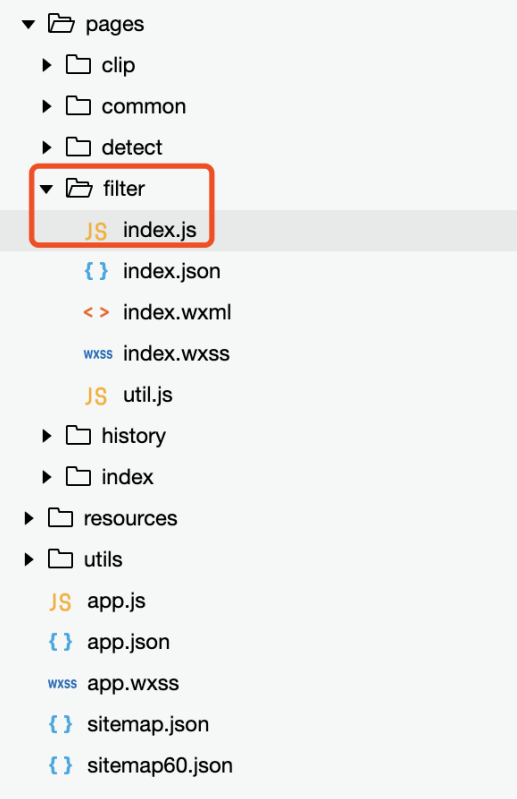
```

let totalLength = data.length;
let radius = kernel.length >> 1;
let channels = [0, 0, 0, 0];
let weight = 0;
for (let index = 0; index < kernel.length; index++) {
  let cur = pointIndex - (radius - index) * width * channels.length;
  if (cur < 0) {
    continue;
  } else if (cur > totalLength) {
    break;
  } else {
    channels.forEach((channel, channelIndex) => {
      channels[channelIndex] += data[cur + channelIndex] * kernel[index];
    });
    weight += kernel[index];
  }
}
return channels.map(channel => {
  return channel / weight;
});

```

这段代码是对图像进行纵向高斯模糊。

再将下面代码粘贴到 **pages/filter/index.js** 的 `handleSmoothTap` 函数内（也就是放到对应的注释里，注释标有 `do something`）：



```

157 // 对纵向像素进行处理，并实现 smoothX 函数
158 // do something
159
160 // 处理完成将平滑后的人脸像素输出到画布
161 // 类似 handleOldTap 最后操作
162 // do something
163
164 for (let x = 0; x < width; x++) {
165   for (let y = 0; y < height; y++) {
166     let pointIndex = (x + y * width) * 4;
167     let channels = smoothX(
168       pointIndex,
169       gKernel,
170       y * width * 4,
171       ((1 + y) * width - 1) * 4,
172       originImageData.data
173     );
174     channels.forEach((channel, index) => {
175       smoothData[pointIndex + index] = channel;
176     });
177   }
178 }
179 for (let x = 0; x < width; x++) {
180   for (let y = 0; y < height; y++) {
181     let pointIndex = (x + y * width) * 4;
182     let channels = smoothY(pointIndex, gKernel, width

```

```

// 通过 smoothX 函数对原有像素进行横向扫描，存储处理后像素于 smoothData
for (let x = 0; x < width; x++) {

```

```

for (let y = 0; y < height; y++) {
  let pointIndex = (x + y * width) * 4;
  let channels = smoothX(
    pointIndex,
    gKernel,
    y * width * 4,
    ((1 + y) * width - 1) * 4,
    originImageData.data
  );
  channels.forEach((channel, index) => {
    smoothData[pointIndex + index] = channel;
  });
}
}

// 对纵向像素进行处理, 并实现 smoothY 函数
for (let x = 0; x < width; x++) {
  for (let y = 0; y < height; y++) {
    let pointIndex = (x + y * width) * 4;
    let channels = smoothY(pointIndex, gKernel, width, smoothData);
    channels.forEach((channel, index) => {
      smoothData[pointIndex + index] = channel;
    });
  }
}

// 将平滑后的人脸像素输出到画布
wx.canvasPutImageData({
  canvasId: "canvas",
  x: 0,
  y: 0,
  width: width,
  height: height,
  data: smoothData,
  success: data => {
    resultImageData = { width, height, data: smoothData };
  },
  fail: e => {
    console.log(e);
  },
  complete: () => {
    // 绘制完成
    this.pending(false);
  }
});

```

除了使用高斯模糊算法对图片的像素进行横轴、纵向处理，在将平滑后的人脸像素输出到画布使用了wx.canvasPutImageData函数

9. 将智能裁剪的照片信息提交到云数据库



使用微信开发者工具打开pages/clip/index.js将下面代码粘贴到第155行处：

```
let { fileID } = await uploadImage(tempFilePath);
let db = wx.cloud.database();
let collection = db.collection('pictures');
if (!collection) {
  throw {
    message: '需创建集合 pictures'
  };
}
await collection.add({
  data: {
    origin: this.data.fileID,
    output: fileID,
    createTime: new Date().getTime()
  }
});
```

这段代码是将照片的fileID、创建时间保存到pictures集合之中，主要用到的是给云数据库指定结合新增记录的知识。

10. 获取云端存储的图片列表



使用微信开发者工具打开pages/history/index.js将下面代码粘贴到第19行处。

```
let { data } = await collection
  .orderBy("createTime", "desc")
  .limit(20)
  .get();
this.setData({
  list: data.reduce((list, { origin, output }) => {
    list.push(origin);
    list.push(output);
    return list;
  }, [])
});
```

这段代码的意思是按照照片的创建时间进行降序排序，选前20张。