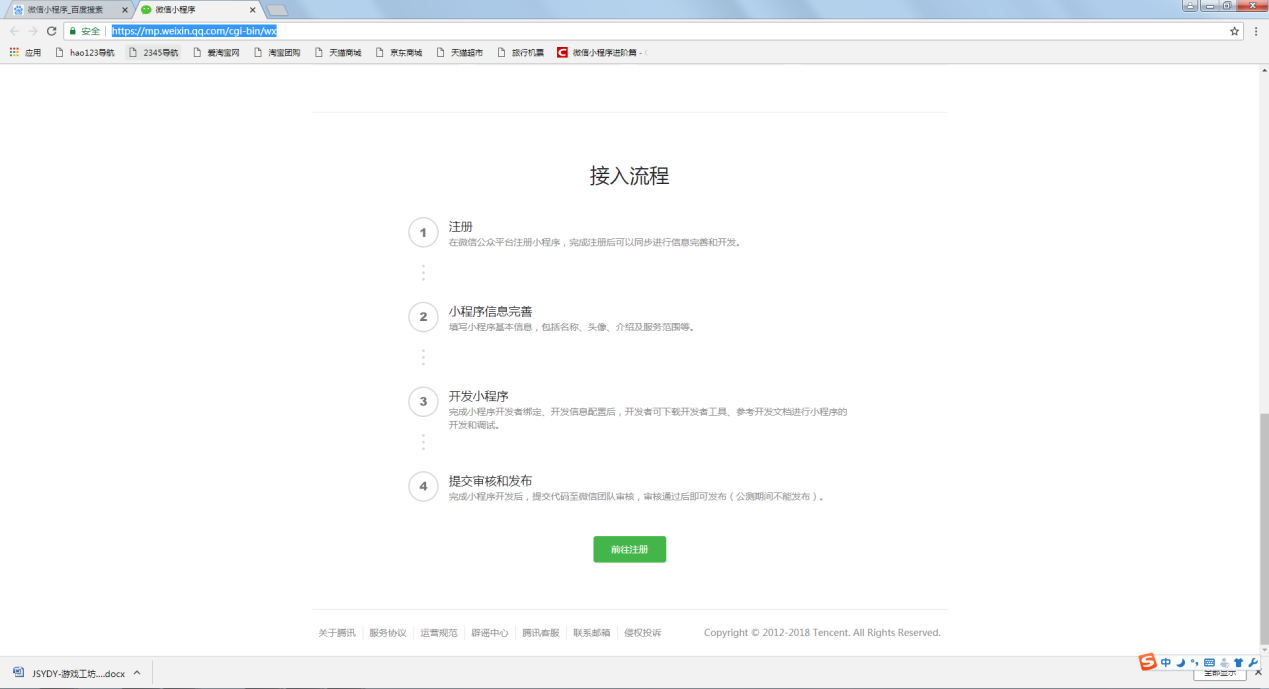
# 单元1微信web开发者工具

## 任务1.1 注册微信小程序账号

1.进入微信公众平台小程序模块：https://mp.weixin.qq.com/cgi-bin/wx。

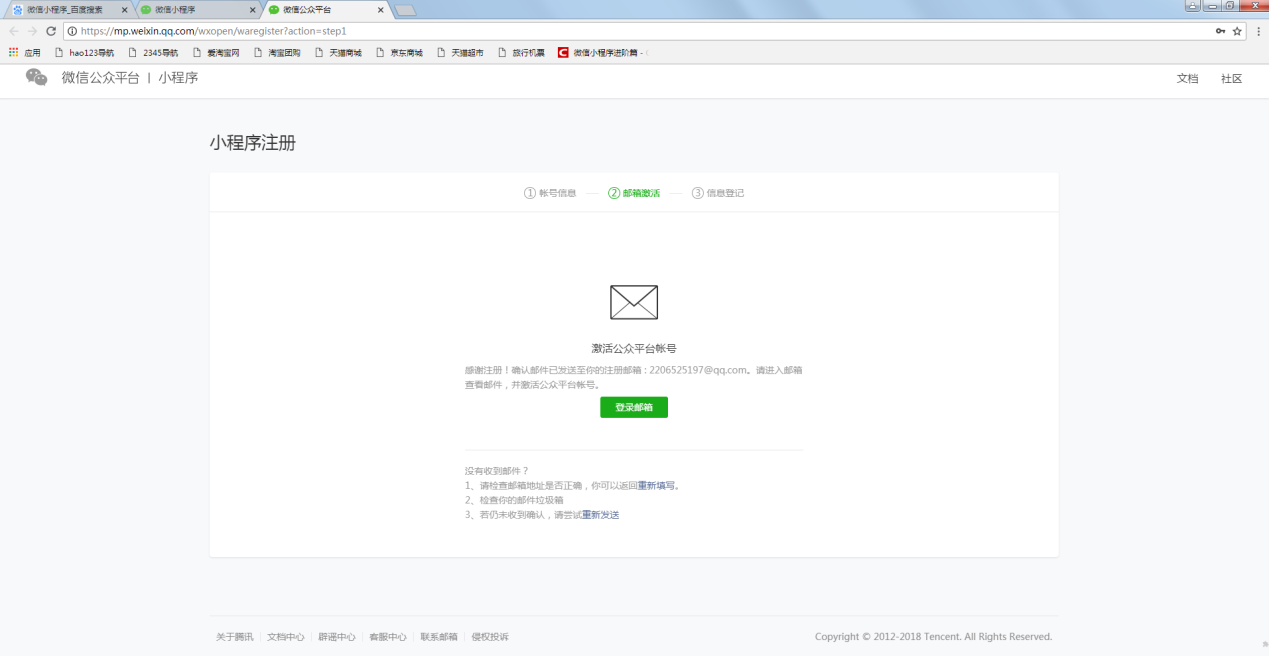
在页面最下方点击前往注册按钮。



2.根据页面要求，填写完注册信息。



3.根据页面提示，前往注册邮箱进行激活公众平台操作。

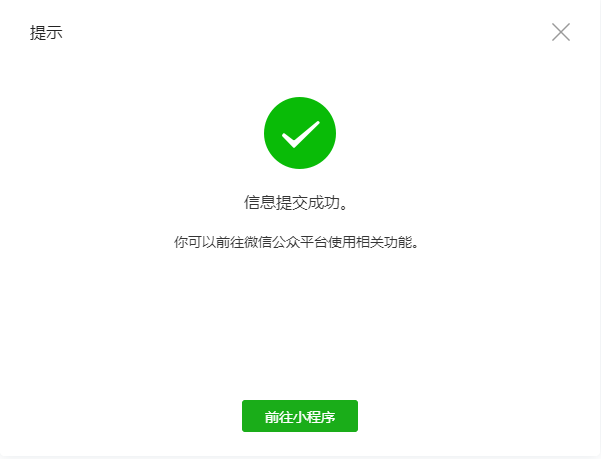


4.激活邮箱以后，进入信息登记页面。选择合适的主体类型，进行相关信息的登记。后续操作以“个人”为主体类型进行演示。



点击页面最下方的继续按钮。

5.根据提示，完成账号的申请。点击前往小程序按钮进入小程序配置页面。

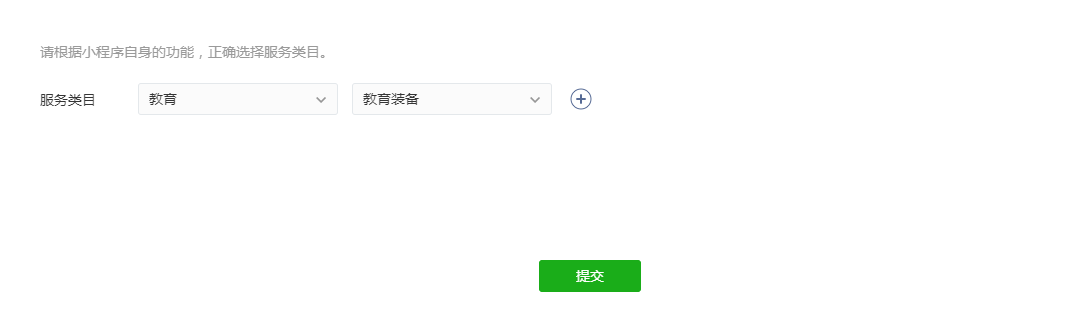


6.点击填写按钮进入填写小程序信息页面。



7.根据页面提示，完成信息的填写。





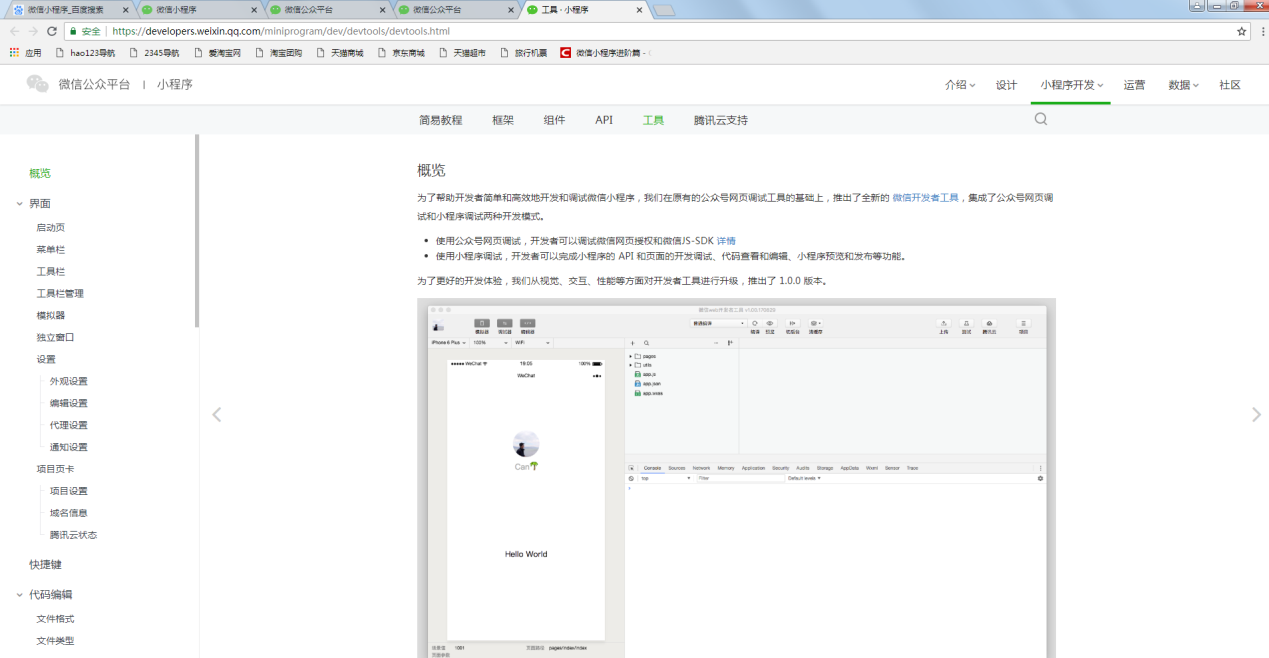
8.点击提交，就完成了小程序信息的录入。

## 任务1.2 开发工具

1.下载开发工具.



进入页面，继续点击微信开发者工具按钮

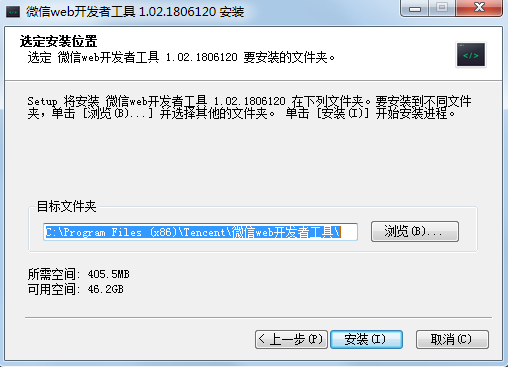


选择适合你机型的版本下载工具。



2.安装开发者工具。

点击运行刚下载的exe文件，根据提示一步步执行，最后点击安装按钮。



耐心等待以后，就提示安装完成了。

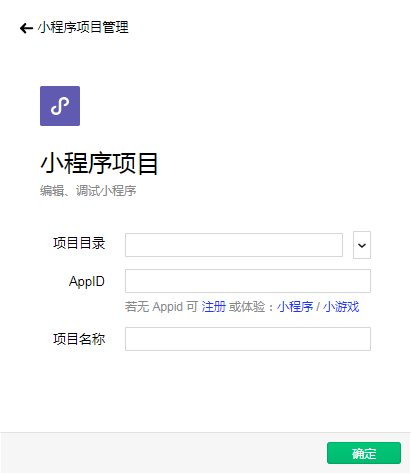


## 任务1.3新建小程序

1.选择小程序项目。



2.根据提示填写项目信息。



**注意：AppID可以直接选择小程序/小游戏选项，但不能进行小程序发布的操作。**

查看AppID，进入开发者公众平台，点击如下图所示按钮。



下图所示就是我们需要用的AppID。

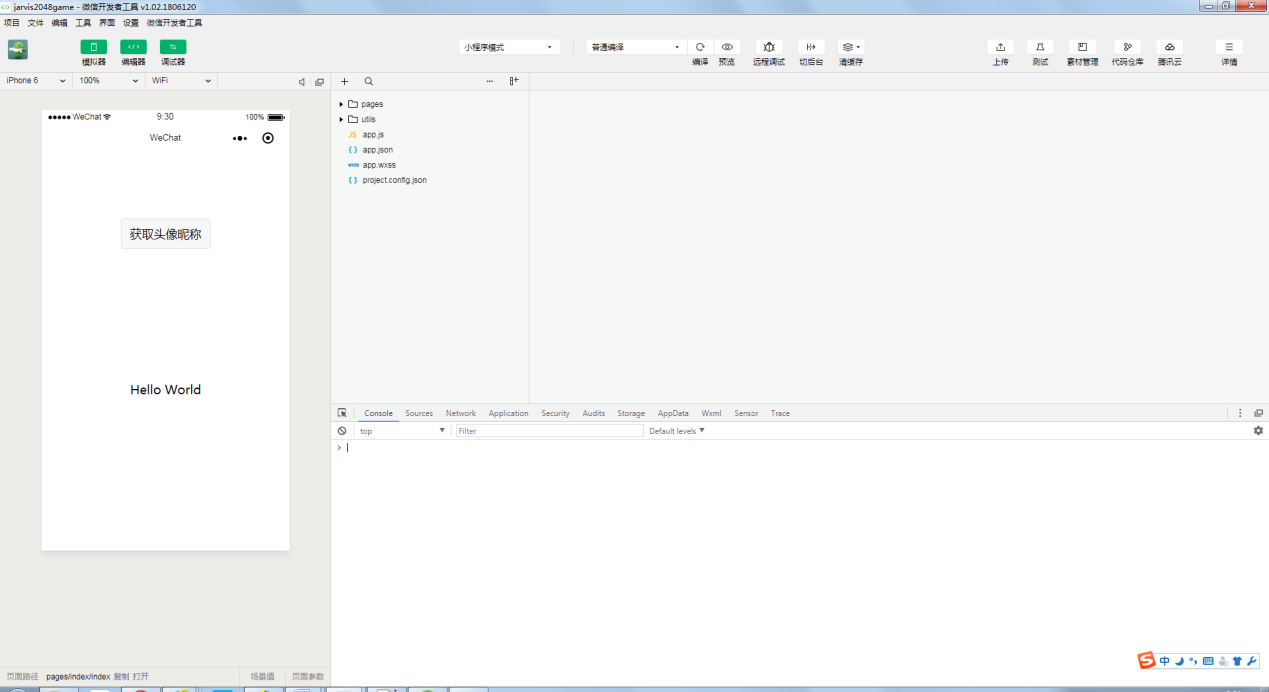


3.当填写完所有信息之后，会出现下图所示样子。



**注意：勾选并且选择简历普通快速启动模板。**

4.点击确认按钮，新建的第一个项目就完成了。



# 单元2 需求分析

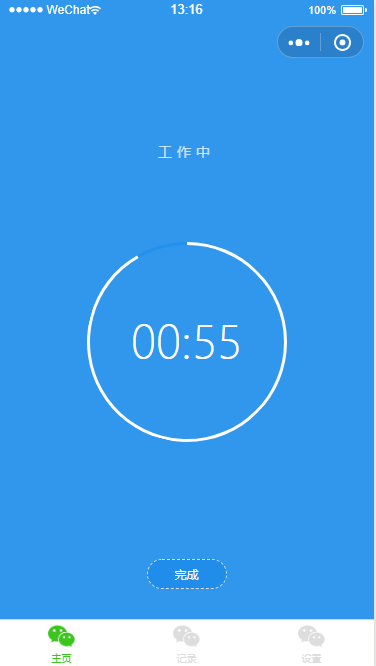
## 任务2.1 功能介绍

1. 开始界面

1.1 开始界面的中央处有倒计时钟，分布在其下的为“工作”与“休息”的两个按钮，点下任一个按钮即可触发响应的倒计时进行计时工作。

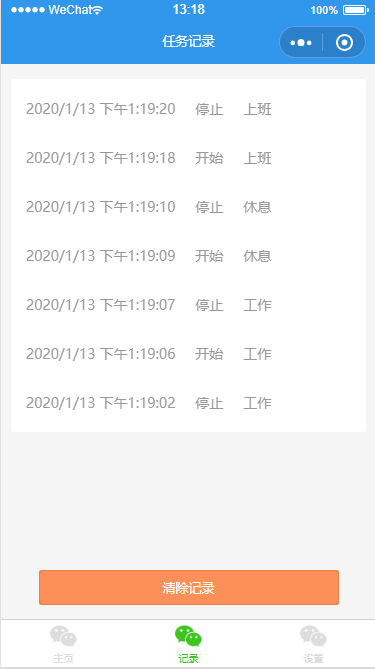
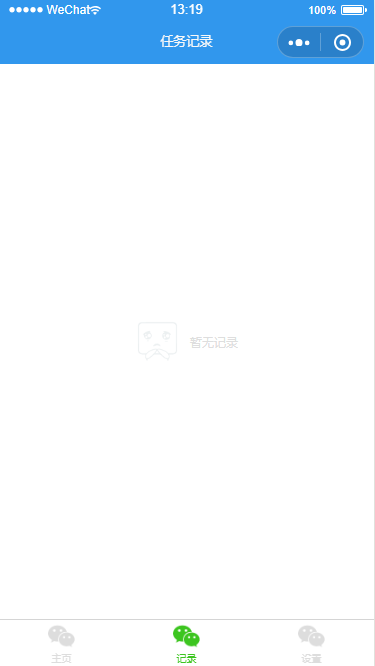
1.2 在按钮下发有一个自动聚焦的输入框，在此处输入想要专注的任务，然后点击上方的按钮即可进行专注工作。

1.3最下方为选项卡，第一个对应的即为开始界面，第二个位记录界面，第三个为设置界面。选中图标会跳转到相应的界面同时图标会变成绿色。

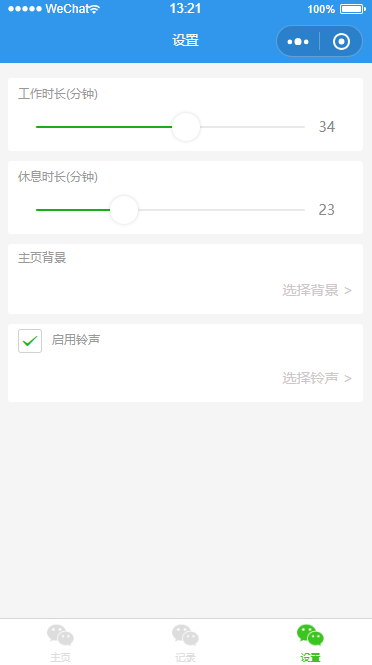
1. 记录界面

会显示“时间段+任务名“，时刻提醒自己做了些什么，底部正中央有个“清除记录”的选项，点击后会清空所有记录。

1. 设置界面

工作时长和休息时长可调，最上方有两个滑块，滑动即可进行时间调节。



# 单元3 开始界面

## 任务3.1 页面设计

1. 整理初始代码，删除部分文件及代码。

1.1 打开app.js文件，删除自带代码，只保留以下代码。

//app.js

App({

})

1.2 删除logs文件包。

1.3 打开app.json文件，删除"pages/logs/logs"，保留以下代码。

{

"pages":[

"pages/index/index"

],

"window": {

"backgroundTextStyle": "light",

"navigationBarBackgroundColor": "#3197ed",

"navigationBarTitleText": "",

"navigationBarTextStyle": "white"

}

}

1.4 删除index.wxml和index.wxss中所有代码。

1.5 打开index.js文件，删除自带代码，只保留以下代码。

//index.js

//获取应用实例

const app = getApp()

Page({

})

1.6 打开app.wxss文件，修改代码。设置页面字体属性及宽高、背景色等通用属性。

/\*\*app.wxss\*\*/

view{

box-sizing: border-box;

}

page {

height: 100%;

}

.container,input,button {

font: 14px "Helvetica Neue", "Hiragino Sans GB", "Microsoft YaHei", Arial, sans-serif;

}

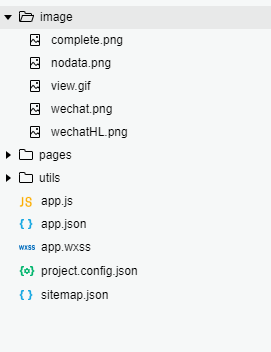
.container {

height: 100%;

background-color: #f5f5f5;

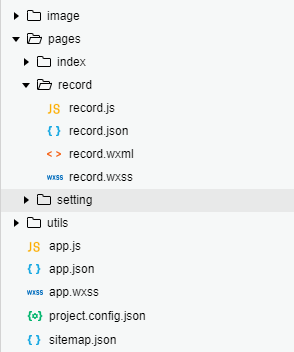
}

1.7 将附件中的imgage文件夹拷贝进项目，放在与pages同级目录下。

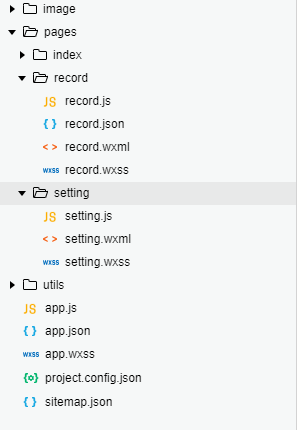


2.构造底部tab页效果

2.1新建pages/record目录，在该目录下新建record页面。项目结构如下图：



2.2 相同的办法新建setting页面。项目结构如下图所示：



2.3打开app.json文件，添加tab页设置，顺序添加如下代码：

"tabBar": {

"color": "#dddddd",

"selectedColor": "#3cc51f",

"borderStyle": "black",

"backgroundColor": "#ffffff",

"list": [

{

"pagePath": "pages/index/index",

"iconPath": "image/wechat.png",

"selectedIconPath": "image/wechatHL.png",

"text": "主页"

},

{

"pagePath": "pages/record/record",

"iconPath": "image/wechat.png",

"selectedIconPath": "image/wechatHL.png",

"text": "记录"

},

{

"pagePath": "pages/setting/setting",

"iconPath": "image/wechat.png",

"selectedIconPath": "image/wechatHL.png",

"text": "设置"

}

]

},

3. 开始页的视图层渲染

3.1 修改pages/index/index.wxml文件。

<view class="container timer {{isRuning ? 'timer--runing': ''}}">

  <view class="timer\_main">

     <view class="timer\_time-wrap">

        <view class="timer\_progress\_mask"></view>

        <view class="timer\_progress timer\_left">

          <view class="timer\_circle timer\_circle--left" style="transform: rotate({{leftDeg}}deg);"></view>

        </view>

        <view class="timer\_progress timer\_right">

          <view class="timer\_circle timer\_circle--right" style="transform: rotate({{rightDeg}}deg);"></view>

        </view>

        <text wx:if="{{!completed}}" class="timer\_time">{{remainTimeText}}</text>

        <text

          wx:if="{{isRuning}}"

          animation="{{nameAnimation}}"

          class="timer\_taskName">{{taskName}}{{completed ? '已完成！' : '中'}}</text>

     <image  wx:if="{{completed}}" class="timer\_done"  src="../../image/complete.png"></image>

     </view>

     <input

      type="text"

      placeholder-style="text-align:center"

      class="timer\_inputname"

      bindinput="changeLogName"

      placeholder="给您的任务取个名字吧"/>

  </view>

  <view class="timer\_footer">

   <view

     bindtap="startTimer"

     data-type="work"

     class="timer\_ctrl {{isRuning && timerType == 'rest' ? 'hide' : ''}}" >{{isRuning ? '完成': '工作'}}</view>

   <view

     bindtap="startTimer"

     data-type="rest"

     class="timer\_ctrl {{isRuning && timerType == 'work' ? 'hide' : ''}}" >{{isRuning ? '完成': '休息'}}</view>

  </view>

</view>

3.2 修改pages/index/index.wxss文件。为index页面添加样式引用，添加如下代码：

/\*\*index.wxss\*\*/

.container {

display: flex;

height: 100%;

flex-direction: column;

overflow: hidden;

background-color: #fff;

}

.timer\_main {

position: relative;

display: flex;

flex: 2;

justify-content: center;

align-items: center;

text-align: center;

background-color: #3197ed;

transition: all .5s;

z-index: 1;

padding: 10px 0;

}

.timer\_time-wrap {

position: relative;

display: flex;

justify-content: center;

align-items: center;

width: 150px;

height: 150px;

text-align: center;

transition: all .3s;

}

.timer\_progress {

position: absolute;

top: 0;

width: 75px;

height: 150px;

overflow: hidden;

transition: all .3s;

}

.timer\_progress\_mask {

position: absolute;

top: 0;

left: 0;

bottom: 0;

right: 0;

border: 3px solid #1589eb;

opacity: .5;

border-radius: 50%;

}

.timer\_left {

left: 0;

}

.timer\_right {

right: 0;

}

.timer\_circle {

position: absolute;

top: 0;

width: 150px;

height: 150px;

border: 3px solid transparent;

border-radius: 50%;

transition: all .3s;

}

.timer\_circle--left {

left: 0;

border-left: 3px solid #fff;

border-bottom: 3px solid #fff;

transform: rotate(45deg);

}

.timer\_circle--right {

right: 0;

border-right: 3px solid #fff;

border-bottom: 3px solid #fff;

transform: rotate(-45deg);

}

.timer\_time {

font-size: 40px;

color: #fff;

font-weight: lighter;

transition: font-size .3s;

}

.timer\_taskName {

position: absolute;

top: -100px;

font-size: 14px;

letter-spacing: 5px;

color: #fff;

}

.timer\_done {

width: 64px;

height: 64px;

}

.timer\_inputname {

position: absolute;

bottom: -40px;

width: 100%;

text-align: center;

height: 40px;

padding-left: 10px;

border-bottom: 1px solid #f2f2f2;

color: #999;

}

.timer\_footer {

display: flex;

justify-content: center;

align-items: center;

flex: 1;

padding-top: 40px;

transition: all .3s;

}

.timer .timer\_ctrl {

display: flex;

justify-content: center;

align-items: center;

text-align: center;

font-size: 12px;

color: #fff;

width: 80px;

height: 80px;

margin: 0 20px;

border-radius: 50%;

transition: all .7s;

background-color: #48c23d;

}

.hide {

display: none!important;

}

/\*runing style\*/

.timer--runing .timer\_main {

flex: 1;

}

.timer--runing .timer\_time {

font-size: 45px;

}

.timer--runing .timer\_time-wrap {

width: 200px;

height: 200px;

}

.timer--runing .timer\_progress {

width: 100px;

height: 200px;

}

.timer--runing .timer\_circle {

width: 200px;

height: 200px;

}

.timer--runing .timer\_footer {

flex: 0;

position: absolute;

bottom: 0;

width: 100%;

z-index: 10;

}

.timer--runing .timer\_ctrl {

background-color: #208DEA;

height: 30px;

margin-bottom: 30px;

border: 1px dashed #dedede;

border-radius: 20px;

}

## 任务3.2 逻辑层开发

1 定义全局默认数据变量。

打开app.js文件，在App函数上方定义defaultTime对象，添加如下代码：

const defaultTime = {

defaultWorkTime: 25,

defaultRestTime: 5

}

在App函数里添加onLaunch函数，在小程序启动时，设置默认参数，并且将相应的数值保存到缓存中，添加代码如下:

onLaunch: function() {

let workTime = wx.getStorageSync('workTime')

let restTime = wx.getStorageSync('restTime')

if (!workTime) {

wx.setStorage({

key: 'workTime',

data: defaultTime.defaultWorkTime

})

}

if (!restTime) {

wx.setStorage({

key: 'restTime',

data: defaultTime.defaultRestTime

})

}

}

2.定义页面临时变量，用来控制按钮显示内容展示和圆环显示角度。打开index.js文件在，在Page函数前添加临时变量，代码如下：

const defaultLogName = {

work: '工作',

rest: '休息'

}

const actionName = {

stop: '停止',

start: '开始'

}

const initDeg = {

left: 45,

right: -45,

}

3.添加页面数据绑定变量，在Page函数中添加data对象，与xml文件绑定显示,代码如下：

data: {

remainTimeText: '',//显示文字

timerType: 'work',//定时器类型

log: {}, //记录

completed: false,//是否已完成定时器

isRuning: false,//定时器是否在运行

leftDeg: initDeg.left,//左半圆进展弧度

rightDeg: initDeg.right//右半圆进展弧度

},

4.定义formatTime函数，作用是时间转换，保证都是二位数显示。如将1转换成01,2转换成02，代码如下：

formatTime: function(time, format) {

let temp = '0000000000' + time

let len = format.length

return temp.substr(-len)

},

5.在页面展示的时候，将缓存中的保存的数据提取出来，显示在页面上。在onShow生命周期函数中添加如下代码：

onShow: function() {

if (this.data.isRuning) return //如果计时器在运行，则直接不执行下面代码

//获取缓存中设置的时间信息

let workTime = this.formatTime(wx.getStorageSync('workTime'), 'HH')

let restTime = this.formatTime(wx.getStorageSync('restTime'), 'HH')

this.setData({

workTime: workTime,

restTime: restTime,

remainTimeText: workTime + ':00'

})

},

6.倒计时功能实现及保存操作记录

6.1 添加保存操作记录的saveLog函数，将操作记录顺序保存进缓存中，代码如下：

saveLog: function(log) {

var logs = wx.getStorageSync('logs') || []

logs.unshift(log)

wx.setStorageSync('logs', logs)

}

6.2 为输入框添加数值改变监听事件changeLogName，代码如下：

changeLogName: function(e) {

this.logName = e.detail.value

},

6.3 为任务名称添加呼吸灯闪烁效果动画，代码如下：

startNameAnimation: function() {

let animation = wx.createAnimation({

duration: 450

})

animation.opacity(0.2).step()

animation.opacity(1).step()

this.setData({

nameAnimation: animation.export()

})

},

6.4 倒计时结束时，设置结束标记，恢复圆环进度条，关闭定时器，添加如下代码：

stopTimer: function() {

// reset circle progress

this.setData({

leftDeg: initDeg.left,

rightDeg: initDeg.right

})

// clear timer

this.timer && clearInterval(this.timer)

},

6.5 根据设置的倒计时时间，更新显示时间和圆环进度条显示状态，添加代码如下：

updateTimer: function() {

let log = this.data.log

let now = Date.now()

let remainingTime = Math.round((log.endTime - now) / 1000)

let H = this.formatTime(Math.floor(remainingTime / (60 \* 60)) % 24, 'HH')

let M = this.formatTime(Math.floor(remainingTime / (60)) % 60, 'MM')

let S = this.formatTime(Math.floor(remainingTime) % 60, 'SS')

let halfTime

// update text

if (remainingTime > 0) {

let remainTimeText = (H === "00" ? "" : (H + ":")) + M + ":" + S

this.setData({

remainTimeText: remainTimeText

})

} else if (remainingTime == 0) {

this.setData({

completed: true

})

this.stopTimer()

return

}

// update circle progress

halfTime = log.keepTime / 2

if ((remainingTime \* 1000) > halfTime) {

this.setData({

leftDeg: initDeg.left - (180 \* (now - log.startTime) / halfTime)

})

} else {

this.setData({

leftDeg: -135

})

this.setData({

rightDeg: initDeg.right - (180 \* (now - (log.startTime + halfTime)) / halfTime)

})

}

},

6.6 为开始按钮添加startTimer单击事件，获取到设置的值以后，启动定时器，每秒更新页面显示状态，记录此次操作保存进缓存。添加代码如下：

startTimer: function(e) {

let startTime = Date.now()

let isRuning = this.data.isRuning

let timerType = e.target.dataset.type

let showTime = this.data[timerType + 'Time']

let keepTime = showTime \* 60 \* 1000

let logName = this.logName || defaultLogName[timerType]

if (!isRuning) {

this.timer = setInterval((function() {

this.updateTimer()

this.startNameAnimation()

}).bind(this), 1000)

} else {

this.stopTimer()

}

this.setData({

isRuning: !isRuning,

completed: false,

timerType: timerType,

remainTimeText: showTime + ':00',

taskName: logName

})

this.data.log = {

name: logName,

startTime: Date.now(),

keepTime: keepTime,

endTime: keepTime + startTime,

action: actionName[isRuning ? 'stop' : 'start'],

type: timerType

}

this.saveLog(this.data.log)

},

# 单元4 记录界面

## 任务4.1 页面设计

1.记录界面的视图层渲染，打开record.wxml文件，添加如下代码：

<block wx:if="{{logs.length}}">

  <scroll-view class="container" scroll-y="true">

   <view class="log panel">

     <view class="log\_item" wx:for="{{logs}}" wx:for-index="$index" wx:for-item="log">

       <text class="log\_start">{{log.startTime}}</text>

       <text class="log\_action">{{log.action}}</text>

       <text class="log\_action">{{log.name}}</text>

     </view>

   </view>

  </scroll-view>

  <view class="clear">

    <button bindtap="switchModal" class="clear\_btn" size="mini" >清除记录</button>

  </view>

</block>

<block wx:else>

  <view class="nodata">

    <image class="nodata\_img" src="../../image/nodata.png"></image>

    <text class="nodata\_text">暂无记录</text>

  </view>

</block>

2.为记录页面添加样式文件，打开record.wxss文件，添加如下代码：

.container{

  padding-bottom: 50px;

  max-height: 92%;

}

.log {

  color: #999;

  margin: 15px 10px;

}

.log\_item{

  padding: 15px 5px;

}

.log\_item text {

  padding-right: 20px;

}

.clear{

  position: absolute;

  bottom: 0px;

  width: 100%;

text-align: center;

}

.clear .clear\_btn {

  margin: 10px;

  height: 35px;

  line-height: 35px;

  color:#fff;

  background-color: #fd8f58;

  border-radius: 2px;

width: 80%;

}

.panel {

background: #fff;

border-radius: 3px;

padding: 5px 10px;

}

.nodata{

  display: flex;

  justify-content: center;

  align-items: center;

  height: 100%;

}

.nodata\_img{

  display: block;

  width: 45px;

  height: 45px;

}

.nodata\_text{

  color:#dedede;

  font-size: 12px;

  padding-left: 10px;

}

3.为记录界面添加业务逻辑代码

3.1 打开record.js文件，在data对象中添加页面绑定数据，添加代码如下：

logs: []

3.2 封装获取缓存操作记录的getLogs函数，添加如下代码：

getLogs: function() {

let logs = wx.getStorageSync('logs')

logs.forEach(function(item, index, arry) {

item.startTime = new Date(item.startTime).toLocaleString()

})

this.setData({

logs: logs

})

},

3.3 在页面打开的生命周期函数中，调取getLogs函数，添加如下代码：

onShow: function() {

wx.setNavigationBarTitle({

title: '任务记录'

})

this.getLogs()

},

3.3 添加清除缓存记录的clearLog函数，添加如下代码：

clearLog: function(e) {

wx.setStorageSync('logs', [])

this.getLogs()

wx.showToast({

title: '清除成功',

icon: 'success',

duration: 2000

})

}

* 1. 未清除记录按钮添加switchModal绑定事件，代码如下：

switchModal: function() {

wx.showModal({

title: '提示',

content: '是否清除记录？此操作不可恢复!',

success:(res)=> {

if (res.confirm) {

console.log('用户点击确定')

this.clearLog();

} else if (res.cancel) {

console.log('用户点击取消')

}

}

})

},

# 单元5 设置界面

## 任务5.1 页面设计

1. 为设置界面的视图层渲染，打开setting.wxml文件，添加如下代码：

<view class="container">

  <view class="section panel">

   <text class="section\_title">工作时长(分钟)</text>

   <view class="section\_body">

   <slider

     bindchange="changeWorkTime"

     show-value="true"

     min="1"

     max="60"

     value="{{workTime}}"

     left-icon="cancel"

     right-icon="success\_no\_circle"/>

   </view>

  </view>

  <view class="section panel">

   <text class="section\_title">休息时长(分钟)</text>

   <view class="section\_body">

   <slider

     bindchange="changeRestTime"

     show-value="true"

     min="5"

     max="60"

     value="{{restTime}}"

     left-icon="cancel"

     right-icon="success\_no\_circle"/>

   </view>

  </view>

  <view class="section panel">

   <view class="section\_title">

     <text>主页背景</text>

   </view>

   <view class="section\_body">

     <text bindtab="" class="section\_tip">选择背景 > </text>

   </view>

  </view>

  <view class="section panel">

   <view class="section\_title">

     <switch class="section\_check" type="checkbox" size="mini" checked bindchange="switch1Change"/>

     <text>启用铃声</text>

   </view>

   <view class="section\_body">

     <text bindtab="" class="section\_tip">选择铃声 > </text>

   </view>

  </view>

</view>

1. 为设置界面添加样式文件，打开setting.wxss文件，添加如下代码：

.container{

  padding:15px 10px;

}

.section{

  margin-bottom: 10px;

}

.section\_title{

  font-size: 12px;

  color:#999;

}

.section\_check{

  margin-right: 5px;

}

.section\_tip{

  display: block;

  text-align: right;

  color: #CEC9C9;

  padding-bottom: 10px;

}

.section\_body{

  margin-top: 15px;

}

.panel {

background: #fff;

border-radius: 3px;

padding: 5px 10px;

}

3.为设置界面添加业务逻辑代码

3.1 在页面显示的生命周期函数中，添加获取缓存中的设置参数，并赋值给页面显示。添加代码如下：

onShow: function() {

wx.setNavigationBarTitle({

title: '设置'

})

this.setData({

  workTime: wx.getStorageSync('workTime'),

  restTime: wx.getStorageSync('restTime')

})

},

3.2 分别为工作时长和休息时长添加滑块变化值监听函数：将滑块获取到的值保存进缓存中，方便开始界面参数显示，添加代码如下：

changeWorkTime: function(e) {

  wx.setStorage({

    key: 'workTime',

    data: e.detail.value

  })

},

changeRestTime: function(e) {

  wx.setStorage({

    key: 'restTime',

    data: e.detail.value

  })

}