开放API产品手册

2020年9月

目录

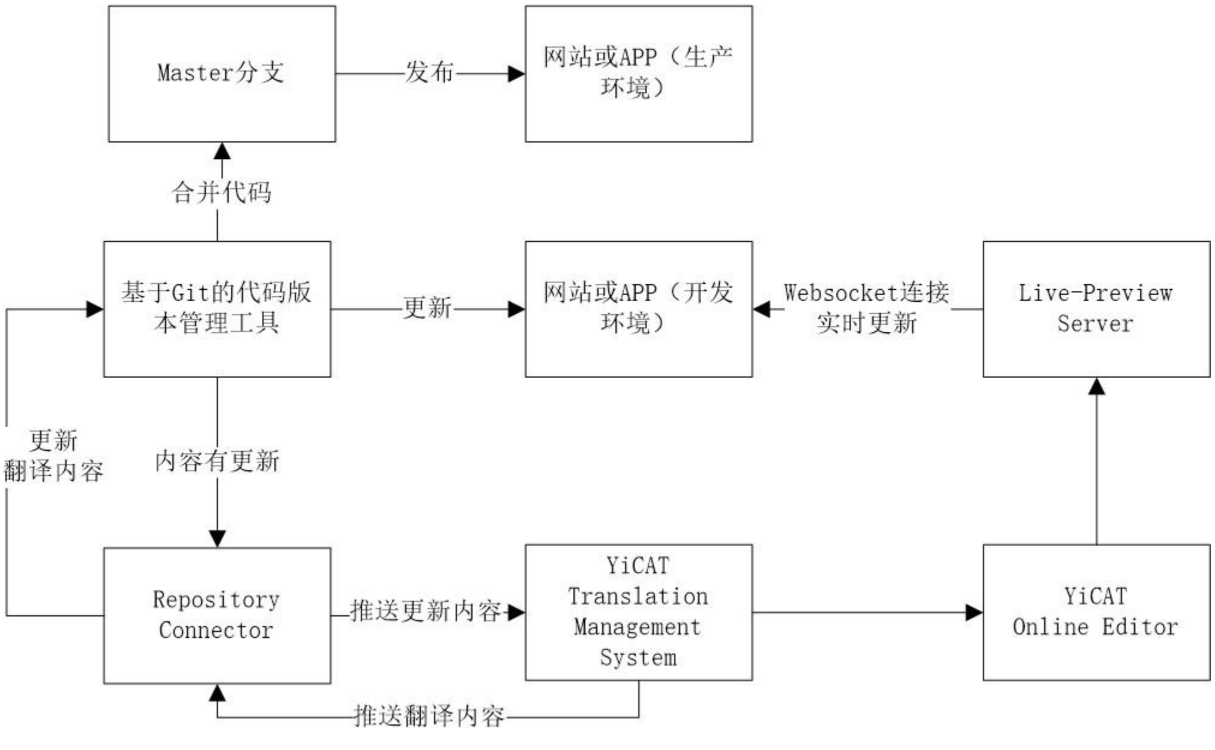
开放API

YiCAT集成器API

随着经济全球化的不断进步，越来越多国内企业需要将网站翻译成多语版本供国外的客户访问和使用，在传统的本地化模式下，需要开发人员将网站中需要翻译的内容抽取出来，发给翻译服务提供商翻译，完成翻译后再将翻译好的内容导回到项目中，开发人员需要就译文内容在网站页面的适配布局显示等与项目经理进行来回沟通。如果内容不断更新和变动，或者后续网站升级，需要翻译的内容持续更新，这个过程会不断重复，效率非常低下，并会导致企业网站内容多语版本严重滞后于本地的版本，影响客户体验。

对于网站、内容管理和APP本地化方案，持续本地化已成为行业主流，相比于传统的翻译方式，持续本地化流程能够在最大程度上更快速、更准确地完成开发，避免重复、易错、低附加值的操作；通过API集成和自动化流程，实现敏捷开发、快速实时翻译和审校，同时可以在模拟器环境下实时预览译文在原文场景下的效果，达到提高网站、内容管理和APP本地化的作业效率的目的。

YiCAT通过开放集成器API接口，可以完成持续本地化，产品实现的方案如下：



以网站开发为例，YiCAT可以帮助提高网站内容翻译效率，它能够提供基于版本管理（如Git, Gitlab, Github, SVN等）的Repository Connector来实现自动将需要翻译的内容上传到YiCAT中进行翻译，在翻译完成以后，可以将翻译好的内容自动推送到指定的目录下，开发人员即可将翻译好的内容发布上线。当出现新的需要翻 译的内容以后，Repository Connector会监控到内容变化，并自动将变化的内容同步到YiCAT中进行翻译，翻译人员仅需翻译新增或者修改的内容。利用YiCAT，翻译过程中还可以重用的翻译语料和术语库，避免重复 翻译、保持翻译一致性；同时YiCAT具备项目管理、译审同步、低错检查等丰富的功能，可以大幅度减少翻译工作中的低效、耗时、重复的劳动，并提供可靠的质量保障，极大提高网站、内容管理维护人员和APP开发人员和翻译人员工作效率，降低翻

译成本。

现在，开发者可以利用我们提供的集成器API（https://api-reference.tmxmall.com）来与自己的网站、内容管理系统集成（CMS）或者APP集成，集成器API包含了添加文件，更新文件，修改句段、导出文件、译文实 时预览等常用接口，通过集成器API和YiCAT可以完成翻译流程中所有操作。

在翻译过程中，翻译人员还可以利Live Preview Server和我们提供的websocket接口实现当前翻译内容的实时预览（In-APP Preview），利用该功能可以帮助翻译人员灵活控制译文的字符串长度，这样可明显减少开发人员在翻译完成后为了适配界面字符显示长度而需要进行额外的样式调整情况。翻译完成确认无误后，开 发人员将翻译的内容同步，进行线上发布使用。

实时预览需要通过集成我们提供的websocket接口来实现，以下是一个JavaScript的demo：

```
  export default class YicatConnector {
    /**
     * 构造函数，接收帐号配置信息和接收新内容的回调函数
     * @param {邮箱帐号} email
     * @param {*} apiKey
     * @param {*} projectId
     * @param {*} updateHandler
     * @param {*} options
     */
    constructor(email, apiKey, projectId, updateHandler, options = {}) {
        this.email = email
        this.apiKey = apiKey
        this.projectId = projectId
        this.updateHandler =updateHandler
        this.debug = options.debug

       this.host = 'i18n.tmxmall.com'
       this.base_url = 'https://' + this.host
       this.base_ws_url = 'wss://' + this.host

       this.connect()
    }

    /**
     * 关闭链接
     */
    close () {
      clearInterval(this.pingTimer)
     if (this.wsInstance) {
       this.wsInstance.onmessage = () => { }
       this.wsInstance.onerror = () => { }
       this.wsInstance.onopen = () => { }
       this.wsInstance.onclose = () => { }
```

```
    this.wsInstance.close && this.wsInstance.close()
    this.wsInstance = null
    }
}

/**
  * 开始连接
  */
connect() {
    this.close()
    this.wsInstance = new WebSocket(this.base_ws_url +`/i18nSegmentWs/${this.email}/${this.apiKey}/${this.projectId}`)

    /**
      * 监听消息并处理
      */
    this.wsInstance.onmessage = (event) => {
         try {
           let msg = JSON.parse(event.data)
           if (this.debug) {
               console.log('国际化插件获取到消息: ', msg)
           }
           if(msg.msgId.indexOf('ping')>-1){
               return
           }
           if (msg.messageType + '' === '15'){
               let tmp = JSON.parse(msg.responseMessage)
               this.updateHandler &&this.updateHandler({
                   TgtLan: tmp.lan,
                   keyPath :tmp.id,
                   tgtText: tmp.tt
               })
           }
       } catch (error) {}
    }

    /**
      * 连接成功，开始心跳检查
      */
     this.wsInstance.onopen = () => {
        if (this.debug) {
           console.log('国际化插件：Socket链接成功')
        }
        setTimeout(() => {
           this.ping()
        }, 4000)
    }
    if (this.debug) {
       console.log('国际化插件：', this.wsInstance)
    }
```

```
}

/**
 * 心跳检查，每10秒连一次，如果状态不正确，执行重连
 */
ping () {
 clearInterval(this.pingTimer)
 this.pingTimer = setInterval(() => {
      let msg =Object.assign({}, defaultModel)
      msg.messageType = 0 // 心跳
      msg.msgId = 'ping-' + new Date().getTime() +Math.random()
      // 发送在正常状态下发送心跳信息
      if (this.debug) {
         console.log('国际化插件：ping...')
      }
      if (this.wsInstance.readyState === 1) {
        this.wsInstance.send(JSON.stringify(msg))
      } else {
         if (this.debug) {
            console.log('国际化插件：需要重连')
         }
         this.connect()
      }
   }, 5 * 1000)
}

/**
 * 用于页面初始化时，对所有文案进行加载一次的接口
 */
fetchAll (lang) {
    return http({
      url:this.base_url+'/cat/i18n/i18nSegment',
      type: 'GET',
      async: true,
      dataType: 'json',
      data: {email:this.email,api_key:this.apiKey,project_id:this.projectId, tgt_lan: lang || 'en-US'},
      headers: {}
   }).then(data => {
      let segmentInfo = (data || {}).i18nSegment ||{}
      // 转换为message map
      let sentences =segmentInfo[lang]
      let msgMap = {}
      sentences.forEach(s => {
         msgMap[s.id] = s.tt || s.st
      })
      return msgMap
   })
   .catch(() => {
```

```
            return {}
        })
    }
}
```