**H5开放设计**

**V2.0**

修订历史

| 版本 | 修订日期 | 修订人 | 修改内容简述 |
| --- | --- | --- | --- |
| V0.9 | 2015-03-06 | 曾理智 | draft version for review |
| V1.0 | 2015-03-11 | 曾理智 | 评审修改: 删除安全部分内容 |
| V1.1 | 2015-03-16 | 齐放 | 1. 将开发者中心、ops调用saf服务，改为了直接编写dao操作数据库 2. 将app获取h5 url的接口，合并成一个，节省客户端与服务端的交互时间。 |
| V1.2 | 2015-03-18 | 齐放 | 1. 数据模型修改：图片、js、css入库； 2. H5版本维护流程修改：创建、更新、审核详情页； 3. H5相关网关接口修改：命名和返回值； 4. 新增有关js api的描述和调用流程； 5. 去掉1.1中“安全”部分内容。 6. 新增对开发者中心的需求关键点的描述。 |
|  |  |  |  |
|  |  |  |  |

# 用例

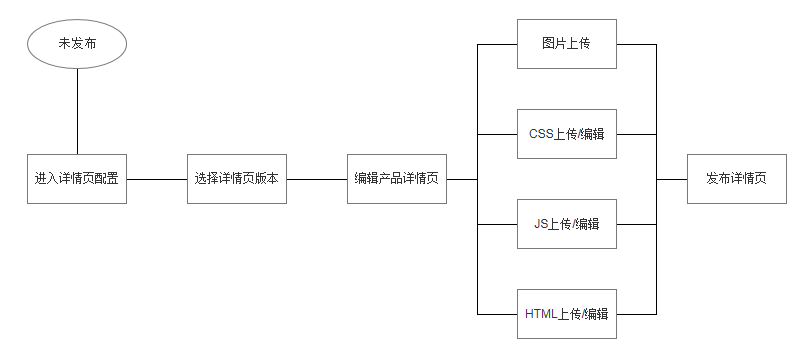




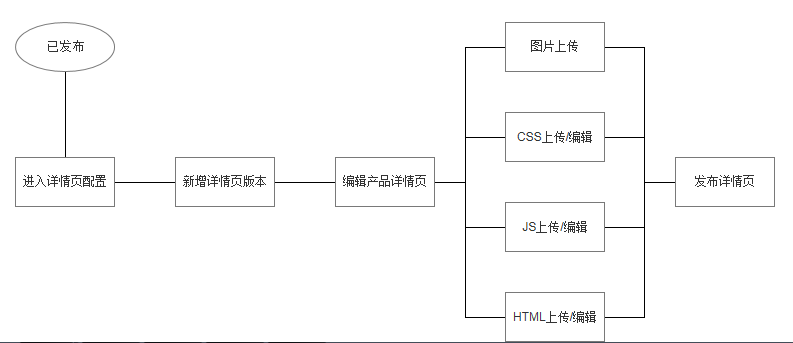


# 开发者中心流程图

未发布：



已发布：



开发者中心需求关键点：

1. 开发者在开发者中心始终只能看到最新的一个版本，若未发布，则可以上传，已发布则只能预览。
2. 所有静态资源（js、css、图片）无版本维护，上传后由开发者自己查看链接，写html代码
3. Html现在暂时不支持编辑，只支持上传
4. 若开发者想编辑已发布的版本，则需新建版本，再上传html和必须的之前未上传的静态资源。
5. 关于Ops的下线处理，将产品的html版本号设置为-1，app查询时返回客户端。
6. 第一期暂时只支持web页面预览，app预览和操控暂不考虑。

# 总体架构



1. ~~建议对服务请求及响应采用数字签名的方式验证来源合法性及防篡改，因此现阶段可以暂不引入开放代理(引擎)，二期添加websocket长连接时，可以再添加开放代理，用于安全代理、连接管理及请求路由。~~
2. 对于H5代码，一期先采取简单的方案，要求开发者先将图片、JS上传到京东JFS服务器，然后在页面代码引用URL，提交完整的html页面代码；微联App使用时，通过JFS url获取到页面代码做渲染展现。 二期再支持上传本地开发包（相对路径引用图片等资源），后端部署H5引擎服务器，生成完整的html代码供app浏览器渲染展现

# 数据模型



1. 新增product\_addtional\_info表，与product表的扩展包，与product表为一一对应关系，
2. 根据product中template\_type, 选择native或h5模板。默认为native，新建h5的时候，将类型改为h5。
3. 新增h5\_template 表，一个h5版本为一条记录。新增一个h5版本时，状态为未发布，版本号为之前最大版本号加一。
4. 详情页审核通过，将h5\_template 表的status状态置为发布的同时，更新product\_addtional\_info中的template\_version。
5. 详情页下线，更新product\_addtional\_info中的template\_version为-1。
6. 新增h5\_resource表，保存开发者上传的静态资源连接，type标识资源类型：js or css or 图片。
7. h5\_template和h5\_resource表中的name和description都用于让开发者标识和备注上传资源的内容。

# 时序图

## JOS授权时序图



与JOS协商定制流程，用户点击确认即可完成授权，不需要重新输入用户名及密码

## 时序图—进入详情编辑页



## 时序图—新增详情页版本



## 时序图—编辑详情页



说明：Jfs的key，使用productId/hash(version+filename)，避免不同产品的不同版本的key有冲突，导致误覆盖

## 时序图—web预览



## 时序图—发布详情页



## 时序图—App加载详情页



1. 模板文件通过京东JFS存储
2. 若模板下线，返回版本号-1

## 时序图—H5调用网关接口



# 服务接口

## 获取包含H5信息的设备快照（进入设备详情页时调用）

|  |  |
| --- | --- |
| 项目 | 描述 |
| 请求方法 | POST |
| 请求URL | http://gw.smart.jd.com/f/service/getStreamsAndH5Info |
| Cookie | \_jsmart\_=U6LN5GTF2Y6CRAM7DKME7KEQFREO5JUGC5VZC4I; Path=/ |
| 请求参数（JSON） | {  "feed\_id": 214739,  “digest”: “-1618480765”  } |
| 返回（JSON） | {"status":0,  "error":null,  "result":"{  \"device\":{  \"access\_key\":\"3dc13543c48163b5d3fa8d7a582d545e\",  \"active\_time\":\"2014-06-19 20:01:27\",  \"device\_id\":\"12345678\",  \"device\_name\":\"jk3\_195403\",  \"feed\_id\":214733,  \"status\":\"0\"  },  \"product\":{  \"p\_description\":\"jk3\",  \"p\_img\_url\":\"\",  \"pro\_type\":-2,  \"product\_id\":12835,  \"product\_name\":\"jk3\",  \"type\_desc\":\"健康类\",  \"type\_name\":\"健康类\"},  \"template\_type\":1,  \"streams\":[{  \"at\":\"\",  \"current\_value\":\"\",  \"master\_flag\":1,  \"stream\_id\":\"sport\",  \"stream\_name\":\"运动数据\",  \"stream\_type\":1,  \"units\":\"\",  \"value\_des\":\"\"  }  ]  }"  \“h5\”:{  \“url\”: \"http://jfs.jd.com/smart/h5/{product\_id}/hash{version+file}\"  \“version\”: 2  }  }”  } |
|  |  |

# 附录: JS API 1.0

**一．JS SDK API:**

1) wl.**getSnapshot**( // 获取设备快照接口

function (res) {

// 执行后的回调, ,开发者自行判断成功与否，及处理数据显示

});

返回数据格式：

{

"status":0, // 返回状态码: 0为成功；非0为失败，对应error字段

"error":null, // 错误信息

"result":"{

"streams": [ // 请求返回的设备快照

{

"stream\_id": "switch", // 控件受控key

"current\_value": 1 // 控件受控value

},

{

"stream\_id": "light",

"current\_value": “on”

}

],

"digest": "-1618480765",

"status":"1" // 设备在线状态：1为在线，0为不在线

}"

}

2) wl.**controlDevice**( // 控制设备接口

{

"command": [

{

"stream\_id": "switch",

"current\_value": "1"

}

]

},

function (res) {

//执行后的回调, ,开发者自行判断成功与否，及处理数据显示

});

返回数据格式：

{

"status": 0, // 返回状态码: 0为成功；非0为失败，对应error字段

"error": { // 错误信息

"errorCode": "",

"errorInfo": ""

},

"result": "{

"control\_ret": "控制成功",

"streams": [ // 同步返回的设备快照，为空则用wl.getSnapshot再取一次

{

"stream\_id": "switch",

"current\_value": 1

},

{

"stream\_id": "light",

"current\_value": "on"

}

],

"digest":"1618480765"

}"

}

3) wl.**getNetworkType**( // 获取APP的网络状态

function (res) {

var networkType = res.networkType; // 返回网络类型2g，3g，4g，wifi

});

二． **JS SDK用法**：

1. 定义全局js sdk 句柄: var wl;
2. 在开发者的html页面中, 加入js sdk监听，并初始化句柄。

document.addEventListener('WLJSBridgeReady', function onReady(ev) {

wl = ev.bridge;

}

1. 调用wl.getSnapshot 及wl.controlDevice 方法来获取设备快照，及控制设备，并刷新html页相应数据

可调用wl.getNetworkType来获取APP的网络状态等native功能。