sgtcloud-html5-sdk用户指南

www.sgtcloud.cn

Published
with GitBook

目錄

sgtcloud-html5-sdk

介绍

sgtcloud是世界领先的游戏baas平台，通过跨平台的sdk，可以立刻把几十种开箱即用的功能加入到您的游戏中，不仅增加了您游戏的可玩内容，也提高了用户的粘性。如果您对游戏有独创性的想法，更可以通过托管代码的paas方式来快速实现。sgtcloud为从独立开发者到完整游戏开发团队的各种类型用户提供了先进，稳定可扩展的联网功能和完善的控制后台，极大的提高了游戏工业的开发效率。我们致力于把先进的技术应用于构建下一代的联网游戏基础设施，提高整个行业的技术水平，促进产业分工，减少重复投入，建立起游戏研发的新模式。

贡献

如果你有好的意见或建议，欢迎给我们提 issue 或 pull request，为提升 sgtcloud-html5-sdk 贡献力量

概述

简介

A html5 out-of-box sdk for damn cool mbaas www.sgtcloud.cn

目的

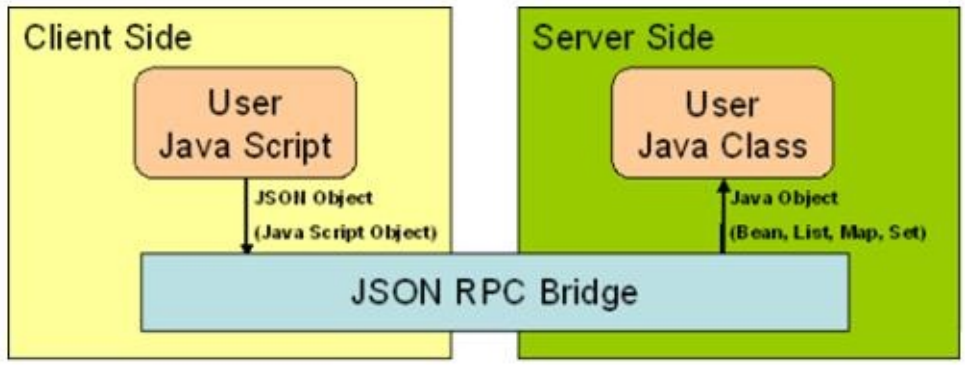
指导合作伙伴的html5客户端工程师开发集成sgt平台开放能力。

范围

所有合作伙伴的html客户端工程师。

技术架构

服务器端是基于Tomcat的微容器
通讯协议是jsonrpc2.0
客户端使用jsonrpc4j作为该协议的实现，使用jackson作为json的序列化和反序列化实现
本sdk仅仅是根据sgt开放平台的业务流程进行了薄封装。



重要概念

callback 回调函数

sdk中的所有服务类方法最后一个参数均可为回调函数,若有则在方法执行完毕后执行.
回调函数中可以包含两个参数

- 第一个参数为方法执行的结果, 值为true或者false
- 第二个参数为方法返回的结果, 值为功能介绍中的return

sgtcloud 平台

包含网关gateway，业务节点node，管理后台console和社交容器opensocial/datacenter构成的mbaas应用系统.

应用标识 appId

应用标识，通过管理后台生成，嵌入到sdk中，每次请求都要携带这个参数，以定位自己的上下文.

网关 gateway

根据应用标识appId和一些其他数据来路由用户到指定的业务节点上.

业务节点 node

每一个业务节点就是一个虚拟的应用容器，可以托管若干应用，也是实现具体交互业务逻辑的地方.

各类 Service

例如UserService, PlayerService等，作为rpc约定的功能集合接口，抽象了一组相关功能方法.
开发者也可以通过自定义的方式来调用服务器端扩展的rpc接口，详细的内容请参看功能介绍.

各类 Entity

例如User，Player等实体类抽象了sgtcloud平台的业务资源.
开发者也可以根据需要自定义一些业务实体，详细的内容请参看功能介绍.

用户 User

sgtcloud平台唯一用户标识，相当于通行证，使用平台功能一定需要先注册一个合法的用户.

角色 Player

大部分功能都需要和至少一个角色关联，所以你需要在使用这些功能前通过一个用户来创建一个角色.

下载安装

你需要 下载 最新版本sgtcloud-html5-sdk

通过github下载

```
git clone https://github.com/sgtcloud/sgtcloud-html5-sdk.git
```

获取下载目录中的 **dist/sgtcloud-html5-sdk.min.js** 文件

安装sgtcloud-html5-sdk

复制 **dist/sgtcloud-html5-sdk.min.js** 文件到你的工程目录,并引用

```
<script src='xxx/sgtcloud-html5-sdk.min.js'></script>
```

或则使用CDN的方式：

```
<script src="http://www.sgtcloud.cn/dist/sgtcloud-html5-sdk.2.0.4.min.js"></script>
```

如何构建

使用 **NPM** 方式

需要

- git
- nodejs
- npm

下载代码

```
git clone https://github.com/sgtcloud/sgtcloud-html5-sdk.git
```

安装依赖

```
npm install
```

构建

```
gulp          //执行编译
gulp -t      //执行测试
gulp -g      //生成jsdoc
gulp --tutorials //启动排行榜教程
```

使用流程

我们假定你处于联网状态,并正确引入了 **sgtcloud-html5-sdk-min.js**
如果需要进行服务器后台操作,请进入后台管理控制中心

初始化（重要）

设置应用标识

```
sgt.init({
    appId: 'html5_demo2015',     //应用标识
    channelId: ''                //渠道Id
});
```

初始化成功之后,就可以使用 sgt 的各种接口了

创建User实例

```
var user = new sgt.User();
user.userName = 'xxx';
user.nickName = 'xxx';
user.password = 'xxx';
```

使用UserService服务

快速登录

```
sgt.UserService.quickLogin(function(result, data) {

});
```

快速登录是sdk中最简单的登录方法，它仅需要一个回调函数即可。
第一次使用方法会随机创建用户名和密码进行登录，并存储在浏览器缓存中，保证下次调用此方法时能继续使用缓存中的帐号。
如果浏览器缓存中找不到帐号信息，则会重新创建

注册User

```
sgt.UserService.regist(user, function(result, data) {
    if (result) {
        //注册成功
    } else {
        //注册失败
    }
});
```

任何一个Service,都允许传入一个回调函数,将在服务器操作成功后执行,

回调函数拥有两个参数:

result 值为true/false,代表操作成功与否

data data代表服务器返回的数据, 请参考jsdoc的功能介绍

登录User

```
sgt.UserService.login(userName, password, function(result, data) {
    if(result) {
        //登录成功
    } else{
        //登录失败
    }
};
```

路由服务

路由服务不需要开发者手动配置
在完成注册或登录服务之后会自动执行路由服务
sdk上下文中会记录用户信息以及服务器信息
相当于下面代码

```
sgt.context = {
    user: {},    //用户信息
    server: {}    //服务器信息
}
```

提示

其他服务动态依赖路由服务
因此在使用其他服务之前首先要进行注册/登录

功能介绍

SgtApi是根据microservice的理念进行架构的

- SgtApi（或者你可以使用sgt这个别名）是全局对象，所有api都在这个namespace之下
- 每个特定的业务抽象成一个xxxService的单例对象
- 每个service内的包含着一组内聚的业务函数
- 每次函数调用都能转换为一次jsonrpc的调用

用户和角色

这是两个核心的业务模块，绝大部分的业务模块是依赖于这两个业务模块的使用。

- 用户是指全平台唯一的业务实体，相当于其他网站的通行证的概念，在不同的应用（appid）和不同的服务器（node）上具有唯一性
- 角色是指在进入应用之后，在特定的服务器上创建的应用数据实体，在特定的应用（appid）和特定的服务器（server/node）中是唯一的
- 一个用户可以在任意的应用和服务器上创建多个角色，两者是一对多的关系

下面我们将介绍核心的接口以及服务

用户 - UserService

用户相关业务接口

使用逻辑

这个服务主要有两种用法，“快速注册/登录”（一键注册/登录）和“普通注册/登录”路径。

快速注册/登录

调用quickLogin即可完成用户的注册或者登录功能。

- 第一次调用从本地localstorage中存储了随机生成的用户名，用这个用户名和固定的密码去注册（register）一个新用户。
- 第二次调用的时候，从localstorage中获取存储的用户名作为登录凭据去登录（login）。
- 可以通过update相关方法更新用户名和密码来重新给用户更换登录凭据，即userid不变，修改username和password，这个调用会同步更新在localstorage中保存的用户名和密码
- 之后的调用quickLogin，会使用更新过的用户名和密码去登录（login）

普通注册和登录

调用register使用（用户输入的）用户名和密码注册新用户，调用login使用使用（用户输入的）用户名和密码登录。

API说明

sgt.UserService.isMatch(smobile, captcha, callback)

- smobile: string 用户手机号
- captcha: string 用户输入验证码
- callback: Function 回调函数
- return: string 是否验证成功结果

验证手机号和验证码是否匹配

sgt.UserService.login(userName, password, callback)

- userName: string 用户名

- password: string 密码
- callback: Function 回调函数
- return: User 登录后的的user对象

手动登录

sgt.UserService.regist(user, callback)

- user: User user对象
- callback: Function 回调函数
- return: User 注册后的user对象

客户端通过提交user对象完成注册

sgt.UserService.resetPassword(userName, callback)

- userName: string 用户名
- callback: Function 回调函数
- return: null

重置密码发送邮件

sgt.UserService.saveLeaveInfo(userLeaveInfo, callback)

- userLeaveInfo: UserLeaveInfo 用户留资对象
- callback: Function 回调函数
- return: UserLeaveInfo 用户留资对象

保存用户留资方法

sgt.UserService.sendCaptchaMessage(smobile, appName, callback)

- smobile: string 用户手机号
- appName: string 当前产品名称
- callback: Function 回调函数
- return: boolean true发送成功, false发送失败

发送手机验证码短信

sgt.UserService.updatePasswordByUserName(userName, password, callback)

- userName: string 用户名
- password: string 密码
- callback: Function 回调函数
- return: null

通过用户名更改密码

sgt.UserService.updateUser(user, callback)

- user: User User数据对象
- callback: Function 回调函数
- return: User 更新之后的user

更新用户信息

sgt.UserService.updateUserByUserId(userid, userName, password, email, callback)

- userid: string 用户ID
- userName: string 用户名
- password: string 密码
- email: string 邮箱
- callback: Function 回调函数
- return: boolean true更新成功, false更新失败

更新用户名，密码，邮箱

sgt.UserService.updateUserNameAndPassword(userid, userName, password, callback)

- userid: string 用户名
- userName: string 用户名
- password: string 密码
- callback: Function 回调函数
- return: boolean true更新成功, false更新失败

更新用户名，密码

sgt.UserService.validationToken(userName, token, callback)

- userName: string 用户名

- token: string token令牌
- callback: Function 回调函数
- return: boolean true合法，false不合法

检测token有效性

sgt.UserService.quickLogin(callback)

- callback: Function 回调函数
- return: User 登录后的user对象

快速登录

sgt.UserService.saveLocalStorage(userName, password)

- userName: string 用户名
- password: string 密码

存储或修改当前应用标识的用户名和密码到localStorage中,便于快速登录

sgt.UserService.removeLocalStorage()

删除localStorage中的当前应用标识的用户名和密码

角色 - PlayerService

角色模型

sgtcloud为各种类型的游戏设计了一个通用的结构化（schema）角色模型Player，这个对象中抽象了角色名称，等级，头像，金钱等属性，可以供开发者存储，更新，查询（暂不提供删除），并且保留了一些可以复用的冗余字段，例如装备。适合轻量级的休闲或者解谜等游戏直接使用。 如果是重度游戏，例如角色扮演，策略等类型，角色信息中包含较多自定义结构和属性的，类似于背包，技能等，应该考虑使用角色扩展来存储，更新和查询。如果您的游戏使用的角色数据信息是二进制的，可以使用content字段来保存，但是该字段无法查询，读写性能也较差，不建存储超过1KB的数据。

- 角色等价于一个游戏内的游戏角色，一个用户在一个游戏内可以创建并且维护多个角色，拥有不同的id，并且可以维护其对应的在线存档。
- 物理上，角色信息存储在分服以后的那台服务器上，所以当你变更过路由策略以后（例如修改渠道），可能会导致服务器上的角色信息找不到。
- 目前角色实体是一个模版，一些字段可以灵活的重用，如果有更多地自定义数据，暂时可以考虑用在线存档在维护。

SgtApi.PlayerService.create(player, callback)

创建一个角色

```
var player = SgtApi.entityFactory('Player');
player.name = '测试玩家AAA';
player.level = 1;
player.money = 999;

var playerId = null;
SgtApi.PlayerService.create(player, function(result, data) {
    if (result) {   //创建成功
        playerId = data.id; //把角色对象的值赋给playerId
    }
});
```

SgtApi.PlayerService.getPlayerById(playerId, callback)

通过角色id获取角色信息

```
SgtApi.PlayerService.getPlayerById(playerId, function(result, data) {
    if (result) {
        data;
    }
});
```

SgtApi.PlayerService.update(player, callback)

更新角色信息

```
player.name = '测试玩家BBB';
player.level = 2;
player.money = 1000;

SgtApi.PlayerService.update(player, function(result, data){
    if (result) {
        data;
    }
});
```

SgtApi.PlayerService.deletePlayerByPlayerId(playerId, callback)

通过playerId删除角色及相关信息

SgtApi.PlayerService.downloadSave(playerId, callback)

下载存档

SgtApi.PlayerService.getByLastLoginTime(lastLoginTime, start, limit, callback);

根据最后登陆时间查找角色

SgtApi.PlayerService.getByName(playerName, start, limit, callback);

根据角色名查找角色

SgtApi.PlayerService.getByUserId(userId, callback);

根据用户ID查找角色

SgtApi.PlayerService.getFriendsMaxNumber(playerId, callback);

获取指定角色的好友上限

SgtApi.PlayerService.getOneByUserId(userId, callback);

通过用户ID查找其中的一个角色

SgtApi.PlayerService.getPlayerByCustomId(customId, callback);

通过自定义ID获取角色信息

SgtApi.PlayerService.searchPlayerByLastLogin(limit, callback);

随机返回若干个最近登录的player

SgtApi.PlayerService.searchPlayersByLastLoginCondition(lastLoginTime, limit, excludePlayerIds, callback);

根据条件过滤并随机查询若干个最近登录的player

SgtApi.PlayerService.setFriendsMaxNumber(playerId, number, callback);

设置指定角色的好友上限

SgtApi.PlayerService.uploadSave(save, callback);

上传存档

Class: Player

- id 主键 非自增，不能为空
- serverId 服务器
- customId自定义ID
- userId 用户ID (opensocial中)
- name 名字
- gender 性别 [1 男 0 女]

- lastLoginTime 最后登录时间
- level 等级
- vip VIP等级
- money 金钱

Class: PlayerExtra

角色扩展信息公共父类，所有开发者扩展的角色信息要么继承这个类，要么在自己的扩展类中添加playerId字段

- playerId 角色ID

Class: Save

- id 主键
- playerId 角色ID
- lastUploadTime 最后上传时间
- content 存档内容
- downFlag 是否可下载存档标识

角色扩展 - PlayerExtraService

角色扩展业务接口
注意: 一个角色只能对应一个角色扩展

sgt.PlayerExtraService.addPlayerExtra(playerExtra, callback)

添加角色扩展信息

sgt.PlayerExtraService.deletePlayerExtraById(playerExtra, callback)

根据角色ID删除角色扩展信息

sgt.PlayerExtraService.findAll(pageNumber, pageSize, callback)

分页查询所有角色扩展信息列表

sgt.PlayerExtraService.findAllByCondition(condition, pageNumber, pageSize, callback)

根据条件查询角色扩展信息列表，支持分页

sgt.PlayerExtraService.getPlayerExtraById(playerId, callback)

根据角色ID查找角色扩展信息

sgt.PlayerExtraService.getPlayerExtraList(condition, callback)

根据条件查询角色列表

sgt.PlayerExtraService.updatePlayerExtraMap(playerExtra, callback)

修改角色扩展信息

sgt.PlayerExtraService.updatePlayerExtra(playerId, playerExtra, callback)

根据角色ID修改角色扩展信息

公告 - AnnouncementService

公告业务接口

SgtApi.AnnoucementService.getAnnounceByType(type, callback)

通过公告类型获取最新公告

成就 - AchievementService

SgtApi.AchievementService.achieve(playerId, achievementId, callback)

达成成就

SgtApi.AchievementService.complete(playerId, achievementId, callback)

领取成就奖励

SgtApi.AchievementService.excuteAchievementsByType(playerId, achievementId, callback)

通过type提交成就 进度数自动+1

SgtApi.AchievementService.customAchievementsByType(playerId, achievementId, callback)

通过成就type累加指定进度

SgtApi.AchievementService.getAchievementById(playerId, achievementId, callback)

根据成就ID获取成就信息

SgtApi.AchievementService.getAchievementsByType(playerId, achievementId, callback)

通过类型获取指定角色可以进行的任务

SgtApi.AchievementService.getAllAchievements(playerId, achievementId, callback)

获取可用的成就

SgtApi.AchievementService.getAvailableAchievements(playerId, achievementId, callback)

获取指定角色未达成的成就

SgtApi.AchievementService.getCompleteAchievements(playerId, achievementId, callback)

获取指定角色已领取奖励的成就

SgtApi.AchievementService.getDoneAchievements(playerId, achievementId, callback)

获取指定角色已经达成的成就

SgtApi.AchievementService.setAchievementProgress(playerId, achievementId, callback)

提交指定成就进度

SgtApi.AchievementService.setAchievementsProgressByType(playerId, achievementId, callback)

设置指定类型的成就进度

活动 - CampaignService

活动是指在一个角色下所能进行的活动，例如当前已经激活的

可通过时间区间、活动ID等获取活动，也可以通过活动ID、获得详情ID获得活动详情数据；也可以更新活动进度。

SgtApi.CampaignService.getAvailableCampaigns(callback)

获取当前已经激活的活动

SgtApi.CampaignService.getByTimeZone(startTime, endTime, callback)

通过时间区间获取活动

SgtApi.CampaignService.getCampaignById(campaignId, callback)

通过活动ID获取活动

SgtApi.CampaignService.getCampaignDataiByCId(campaignId, callback)

通过活动ID获取活动详情数据

SgtApi.CampaignService.getCampaignDetaiById(campaignId, callback)

通过活动详情ID获取活动详情数据

SgtApi.CampaignService.getCampaignProgress(campaignId, playerId, callback)

获取进度

SgtApi.CampaignService.updateProgress(campaignId, playerId, progress, callback)

更新进度

签到 - CheckinBoardService

签到主要分两种类型，一种是每日签到，即每日只能只能签到一次，成功则返回连续签到次数，如果本日签到过再发起签到会返回0，表示当日已经签到过
还有一种类型是不累计次数的签到，等同于一个在线的计数器，可以获取到两次签到的间隔时间
签到功能需要在后台创建一个指定id的签到板才能使用，暂时请联系后台人员协助创建
每个角色在不同的签到板的签到信息是独立的

SgtApi.CheckinBoardService.checkin(playerId, checkinBoardId, callback)

签到

SgtApi.CheckinBoardService.accumlateCount(playerId, checkinBoardId, callback)

获取最大累计签到数

SgtApi.CheckinBoardService.countinuousCount(playerId, checkinBoardId, callback)

获取连续签到数

SgtApi.CheckinBoardService.getCheckinboardByChekinboardId(checkinBoardId, callback)

通过CheckInBoardId获取签到板实体数据

SgtApi.CheckinBoardService.getLastCheckinTime(playerId, checkinBoardId, callback)

获取最后签到时间

SgtApi.CheckinBoardService.getRewardByCheckinBoardId(checkinBoardId, callback)

获取奖励

SgtApi.CheckinBoardService.setCheckinTimes(playerId, checkinBoardId, times, callback)

补签

SgtApi.CheckinBoardService.validateCheckin(playerId, checkinBoardId)

判断是否可以签到

SgtApi.CheckinBoardService.getAvailableCheckinBoards(callback)

获取当前可用（有效时间内）的签到板

SgtApi.CheckinBoardService.getAvailableCheckinBoardsByTag(tag, callback)

根据自定义标签获取当前可用（有效时间内）的签到板

SgtApi.CheckinBoardService.getAvailableCheckinBoardsByType(type, callback)

根据类型获取当前可用（有效时间内）的签到板

Class: CheckinBoard

- CheckinboardId 用来和客户端交互数据的标识是CheckinboardId，类似于排行榜的LeaderboardId，在服务器端创建，客户端操作
- name 签到板名称
- type 目前有每日签到CHECKIN_TYPE_DAILY，普通签到CHECKIN_TYPE_NORMAL，日历签到CHECKIN_TYPE_CALENDAR3种类型
- maxCheckinTimes 最大连续签到次数，在服务器端设置过最大签到次数以后，会把连续签到次数重置为0
- 有效开始时间（可选）
- 有效结束时间（可选） 开始时间和结束时间可以只选择一个，或者都不选 若签到板不在有效期内，则抛出异常 *InvalidStateException("*签到板已过期！*")*
- rewards 奖励
- tag 自定义签到标记，可以用来区分不同的签到类型，例如多个每日签到等
- resetTime 签到状态重置时间，表示第二天{resetTime}时间点可以再次签到，默认为0，

零点整。24小时制，取值范围为0--24

日常任务 - DailyTaskService

角色的某种行为可以关联到一个计数器（可以有多个，每个对应一种行为，比如打竞技场，也就是原来的type类型），计数器就一个属性，就是次数/进度，分为每日（有ttl，每天会重置）和累计两个值。 然后客户端可以发送一个动作类型和每日任务到服务器来判断该计数器是否满足任务的状态变更需求。

SgtApi.DailyTaskService.executeTask(taskId, playerId, callback)

提交任务进度，每提交一次，任务进度+1

SgtApi.DailyTaskService.executeTasksByType(type, playerId, callback)

通过type提交任务，每提交一次，任务进度+1 type只能对应1或0个任务

SgtApi.DailyTaskService.addExecuteTasksByType(type, playerId, progress, callback)

增加指定类型的任务进度

SgtApi.DailyTaskService.getDailyTasks(playerId, callback)

获取每天的日常任务

SgtApi.DailyTaskService.getDailyTasksByType(playerId, type, callback)

通过类型获取指定角色可以进行的任务

SgtApi.DailyTaskService.getReward(taskId, playerId, callback)

根据任务ID获取奖励 获取奖励时会先校验完成任务的进度

SgtApi.DailyTaskService.setTaskProgress(playerId, taskId, progress, callback)

提交指定任务进度

SgtApi.DailyTaskService.setTasksProgressByType(type, playerId, progress, callback)

提交任务进度，每提交一次，任务进度+1

Class: DailyTask

日常任务（DailyTask）至少包括的字段：

- id
- type (类型)
- goal（完成任务的总进度数）
- reward (奖励)
- preTaskId(前置任务)
- nextTaskId(后置任务)
- startTime(开始时间)
- endTime(结束时间)
- currentProgress（当前进度）使用计数器的值
- status（当前任务进度状态）
- available（当前任务可用状态，默认可用，1为可用，0为不可用）
- minLevel(限制等级，最低可做该任务的等级)
- updateUnfinished（是否在前置任务没完成之前同时更新后置任务，默认不允许，true允许，false不允许）
- overMaxProcess（是否允许当前进度超越最大进度，默认不允许。true允许，false不允许）
- showDone（是否允许任务完成后仍然显示在列表,默认显示，false为不显示，true为显示）

排行榜 - LeaderBoardService

排行榜功能需要在后台创建一个指定id的签到板才能使用，暂时请联系后台人员协助创建
排行榜中的每条记录需要包含分数和提交分数的角色id。同一个排行榜中一个角色只能有一个分数
提交分数有两种模式，设置一个分数（覆盖原来的分数），提交一个增加值（累计到原来的分数上

SgtApi.LeaderBoardService.addUpLeaderBoardScore(leaderBoardId, playerId, score, callback)

更新分数值

SgtApi.LeaderBoardService.getLeaderBoardByLeaderId(leaderBoardId, callback)

通过排行榜的leaderBoardID获取leaderBoard信息

SgtApi.LeaderBoardService.getLeaderBoardScoreByExample(leaderBoardId, player, callback)

通过排行榜ID和角色ID获取该角色的排行榜（返回集合）

SgtApi.LeaderBoardService.getLeaderBoardScoreByLeaderIdAndPlayerId(leaderBoardId, playerId, callback)

通过排行榜ID和角色ID获取该角色的排行榜

SgtApi.LeaderBoardService.getLeaderBoardScoresByLeaderIdAndPlayerId(leaderBoardId, playerId, callback)

如果我是第一名则返回我和后面4位 如果不是第一名则返回我前面一位+我+后面3位 如果我是最后一名则返回我前面四位+我

SgtApi.LeaderBoardService.getTopLeaderBoardScoreByLeaderId(leaderId, start, limit, callback)

通过排行榜ID和排名获取排行榜集合 start从0开始，第一名的值是0 取前两名的则 start为0，limit为2 取第二名到第五名则start为1，limit为4 即：start的值为排名减1 如果取该排行榜中所有的排名 start和limit的值分别为：0 , -1

SgtApi.LeaderBoardService.submitLeaderBoardScore(leaderBoardId, playerId, score, callback)

提交排行榜数值

Class: LeaderBoard

- id 主键
- name 名称
- appId APPID
- serverId 服务器
- leaderId 自定义排行榜ID
- activityId 活动ID

Class: LeaderBoardScore

- score 分数
- player 角色信息
- index 排位

邮件 - MailService

邮件功能分为邮件发送和邮件查询,需要拥有自身的账号
角色条件下使用，发送邮件需要实例化MAIL，然后设定邮件各种参数属性：例如Title，Content，ID等
同样接收邮件时也需要将这些属性得到。

SgtApi.MailService.sendMail(mail, callback)

发送一封邮件

SgtApi.MailService.receive(start, limit, playerId, status, callback)

接收邮件

SgtApi.MailService.receiveUnread(playerId, callback)

接收未读取的邮件

SgtApi.MailService.readMail(mailId, callback)

阅读邮件/批量阅读邮件

SgtApi.MailService.readAndPickAttachment(mailId, callback)

阅读一封邮件并领取附件

SgtApi.MailService.deleteMail(mailId, callback)

删除封邮件/批量删除邮件

SgtApi.MailService.getReadedAndUnreadedMails(playerId, callback)

获取所有未读和已读的邮件集合

SgtApi.MailService.pickAttachment(mailId, callback)

领取邮件附件

Class: Mail

邮件（Mail)至少包括字段:

- id 主键
- fromId 发送者ID
- fromName 发送者名称
- toName 接收者名称
- toId 接收者ID
- title 标题
- content 内容
- attachment 附件
- status 状态
- sendTime 发送时间
- type 类型
- attachStatus 附件状态，领取（1）和未领取（0，默认）

Boss - BossService

BOSS在游戏中有不同的设定，字段拥有多个，可以设定他的HP，id，name，PRIVATE，PUBLIC等

用户可以攻击私有boss，世界boss等，私有boss是只有自己可以看到和攻击，而世界boss可以在特定时间，特定任务中攻击，这个可以开发者自己设定

SgtApi.BossService.getByBossIdstr(ids, callback)

通过id字符串获取boss数组

SgtApi.getByBossIdint(id, callback)

批量获取boss数据

SgtApi.getByBossId(id, callback)

通过bossId获取Boss实体

SgtApi.attack(boosId, damage, callback)

更新boss血量

SgtApi.getCurrentHP(bossId, callback)

获取boss当前血量

SgtApi.getLastAttackPlayer(bossId, callback)

获取最后击杀人

Class: Boss

- Id 主键
- name 名称
- hp 血量
- type 类型
- intensity 强度

- rewards 奖励

抽奖 - GachaBoxService

抽奖活动是用户在连续登陆或者别的情况下（根据需要设定）所进行的对用户的奖励，根据名称以及奖品品质获得
首先需要获得GachaBox,可以获得当前所有有效的GachaBox，也可以通过名称获得指定的GachaBox。

SgtApi.GachaBoxService.autobalanceDraw(playerId, gachaBoxId, num, callback)

有自动修正的连抽

SgtApi.GachaBoxService.autobalanceDrawQuality(playerId, gachaBoxId, num, quality, callback)

指定初始品质的自动修正连抽

SgtApi.GachaBoxService.autobalanceDrawMaxQuality(playerId, gachaBoxId, num, quality, maxQuality, callback)

指定初始品质和最大品质的自动修正连抽

SgtApi.GachaBoxService.draw(playerId, gachaBoxId, quality, callback)

连续抽奖N次，N为数组qualities的元素个数，一个qualities元素对应一次抽奖

SgtApi.GachaBoxService.getAvailableGachaBox(callback)

获取当前所有有效的GachaBox

SgtApi.GachaBoxService.getGachaBoxByName(gachaBoxName, callback)

获取指定名称的GachaBox

SgtApi.GachaBoxService.getLotteriesByGachaBoxId(gachaBoxId, callback)

获取指定gachaBox的奖品

SgtApi.GachaBoxService.limitDraw(playerId, gachaBoxId, limitQuality, callback)

指定奖品品质总值的连抽

兑换码 - GiftCodeService

礼包/兑换码业务接口

sgt.GiftCodeService.getGiftByCode(code, callback)

- code: string 兑换码
- callback: Function 回调函数
- return: Gift 礼包详情

通过兑换码获取礼包详情

sgt.GiftCodeService.getGifts(callback)

- callback: Function 回调函数
- return: Gift[] 有效的礼包集合

获取有效的礼包

sgt.GiftCodeService.getRecord(playerId, giftId, start, limit, callback)

- playerId: string 角色ID
- giftId: string 礼包ID
- start: number 起始页码
- limit: number 每页显示条目数
- callback: Function 回调函数
- return: GiftRecord[] GiftRecord集合

获取兑换记录， playerId和giftId至少有一个不为空。 playerId为null则返回该礼包的所有记录 giftId为null则返回该角色所有兑换记录

sgt.GiftCodeService.redeem(playerId, giftId, code, callback)

- playerId: string 角色ID
- giftId: string 礼包ID
- code: string 兑换码
- callback: Function 回调函数
- return: string 奖品

兑换并返回奖品

sgt.GiftCodeService.redeemGiftByCodeOverMail(playerId, code, callback)

- playerId: string 角色ID
- code: string 兑奖码
- callback: Function 回调函数
- return: Gift 成功返回gift，失败返回null

兑奖并使用邮件直接发送奖励给角色

sgt.GiftCodeService.redeemOverMail(playerId, giftId, code, callback)

- playerId: string 角色ID
- giftId: string 礼包ID
- code: string 兑换码
- callback: Function 回调函数
- return: null

兑换奖励，奖励通过邮件发送

黑名单 - BlackListService

黑名单是对应用户之间关系的功能,用户可将一些用户添加进自己的黑名单中

在将所选用户加入到黑名单之前，首先需要判断该用户是否在你的黑名单中，如果没有再将该用户加入到黑名单中。

SgtApi.BlackListService.isInBlackList(blacklistId, playerId, fn)

查询黑名单

SgtApi.BlackListService.addPlayerIntoBlackList(blacklistId, playerId, fn)

更新黑名单

反馈 - TicketService

反馈是用在应用中所遇到的问题与后台交流的一种方式，后台从用户的反馈消息中提取标题内容等所需要的信息
用户提交反馈，后台提取信息
当用户提交了反馈之后，后台会得到相应的数据，通过反馈者playerId获取他发起的反馈信息
得到这个反馈集合之后，可得到每条反馈中的信息。

SgtApi.TicketService.getTicketsById(playerId, page, size, status, callback)

通过反馈者playerId获取 自己发起的反馈信息

SgtApi.TicketService.sendTicket(ticket, callback)

提交反馈

好友 - FriendshipService

好友业务接口

SgtApi.FriendshipService.acceptInvite(sendIds, receiveId, callback)

自己作为被邀请者接受好友邀请

SgtApi.FriendshipService.acceptInviteByMail(sendIds, receiveId, mail, callback)

自己作为被邀请者批量确认/接受好友关系，并发送邮件

SgtApi.FriendshipService.getDenied(page, limit, myId, callback)

获取拒绝自己的请求

SgtApi.FriendshipService.getFrindsCount(playerId, callback)

通过playerId获取该角色的好友数量

SgtApi.FriendshipService.getInvite(receiveId, callback)

获取邀请自己的player

SgtApi.FriendshipService.getMyFriends(page, limit, playerId, callback)

获取指定用户的好友(已确认)列表

SgtApi.FriendshipService.getNotConfirm(sendId, callback)

获取还未确认邀请的player

SgtApi.FriendshipService.invite(sendId, receiveId, callback)

批量邀请加好友（状态为未确认

SgtApi.FriendshipService.inviteByMails(sendId, receiveId, mails, callback)

批量邀请加好友，并发送邮件通知

SgtApi.FriendshipService.isMyfriend(myId, otherId, callback)

判断对方是否是自己的好友

SgtApi.FriendshipService.refuse(sendId, receiveId, callback)

拒绝好友申请

SgtApi.FriendshipService.unfriend(playerId, unfriendId, callback)

批量解除好友关系

好友扩展 - FriendshipExtraService

好友关系扩展业务

SgtApi.FriendshipExtraService.getAllMyFriendsAndExt(myPlayerId, key, start, limit, callback)

枚举自己（发起者）的（已验证）好友和key对应的扩展数据,支持分页

SgtApi.FriendshipExtraService.getMyFriendAndExt(myPlayerId, friendId, callback)

获取自己指定好友和扩展数据

SgtApi.FriendshipExtraService.updateAllMyFriendExt(myPlayerId, key, value, callback)

批量修改自己（发起者）所有好友关系扩展数据中key对应的value的值

SgtApi.FriendshipExtraService.updateMyFriendAllExt(myPlayerId, friendId, newExt, callback)

批量修改指定好友关系的扩展数据,中key对应的值将会替换现有value

SgtApi.FriendshipExtraService.updateMyFriendExt(myPlayerId, friendId, key, value, callback)

修改自己（发起者）某个好友关系扩展数据中key对应的value的值

充值 - PurchaseService

充值业务逻辑

SgtApi.PurchaseService.getSupportedStires(callback)

获取服务器支持的支付渠道

SgtApi.PurchaseService.getAvailableChargePoints(playerId, callback)

获取玩家当前可用的充值信息，不包括不可见或达到购买次数限制的充值信息

SgtApi.PurchaseService.getPaymentResult(playerId, transaction, callback)

获取充值结果，该接口负责去第三方查询支付结果或获取第三方回调信息，根据支付结果，来调用游戏业务逻辑支付成功，会调用PaymentCallback.doCallback执行游戏的业务逻辑（如加元宝、积分等）同时，sgt还记录了玩家的充值记录ChargeLog，充值次数等信息

SgtApi.PurchaseService.getTotalChargeCost(playerId, callback)

获取玩家的累计充值金额，若玩家未充值过，则返回0

SgtApi.PurchaseService.isFirstCharge(playerId, customChargePointId, callback)

判断玩家某个计费点是否是首冲

SgtApi.PurchaseService.getChargeTimes(playerId, customChargePointId, callback)

获取玩家某个计费点的充值次数

计费点 - ChargePointService

计费点业务

sgt.ChargePointService.getAllChargePoints(callback)

- callback: Function 回调函数
- return: ChargePoint[]

获取所有计费点

sgt.ChargePointService.getAvailableChargePoints(callback)

- callback: Function 回调函数
- return: ChargePoint[]

获取当前可用的计费点

通知 - NotificationService

通知模块业务

SgtApi.NotificationService.getLatestNotification(playerId, callback)

获取最新状态的通知/获取某时间之后的通知

SgtApi.NotificationService.getLatestNotificationByTime(playerId, time, callback)

根据时间获取最新状态的通知/获取某时间之后的通知

商城 - StoreService

商城业务接口

sgt.StoreService.getDefaultStore(playerId, callback)

playerId:string 角色id
获取默认的商城

sgt.StoreService.getStore(storeId, playerId, callback)

根据商城ID获得商城，包含了商城中的所有物品，但物品购买冷却时间未设置

sgt.StoreService.purchase(playerId, storeId, itemId, amounts, ckret, callback)

playerId: string 角色id
storeId: string 商城id
itemId: string 物品id
amouts: number 购买数量
ckret: boolean 如果为true, 则需要返回一个订单id(即did), 网友一般需要来操作
购买指定的商品，如果使用游戏币购买，那么在购买前需要同步服务器端的游戏币

sgt.StoreService.countStoreOrdersByPlayerId(playerId, storeId, callback)

获取角色在指定商店的购买次数

sgt.StoreService.countItemOrdersByPlayerId(playerId, storeId, itemId, callback)

获取角色购买某个商品的次数

sgt.StoreService.getOrderById(storeId, callback)

查询购买记录

sgt.StoreService.getLastPurchaseTimeMillis(playerId, storeId, itemId, callback)

获取角色最近购买某个商品的时间的毫秒数

sgt.StoreService.updateOrderStatus(did, success, callbackMessage, callback)

支付回调接口，对订单状态进行更新

微信中控 - WxCentralService

要使用微信中控需要两个条件

- 导入微信js-sdk
- 在微信客户端环境中

微信中控服务接口

sgt.WxCentralService.getAccessToken(appId, callback)

- appId: string SGT中的appid
- callback: Function 回调函数
- return: WxResult 含有accessToken的WxResult

获取微信的accessToken，每一小时刷新一次

sgt.WxCentralService.getJSTicket(appId, callback)

- appId: string SGT中的appid
- callback: Function 回调函数
- return: WxResult 含有jsapi_ticket的WxResult

获取微信的jsapi_ticket

sgt.WxCentralService.getSignature(appId, url, callback)

- appId: string SGT中的appid
- url: string 页面url，必须是调用JS接口页面的完整URL。
- callback: Function 回调函数
- return: WxResult 包含签名以及计算参数的WxResult:{ result: 处理正常result有值 {"signature":"签名","timestamp":"计算时用到的时间戳，单位秒","noncestr":"计算签名的随机字符串"},error:"计算失败时返回的错误信息"

获取jsapi 签名，签名用的noncestr和timestamp必须与wx.config中的nonceStr和timestamp相同。

sgt.WxCentralService.getPayOrder(paramModel, callback)

- paramModel: string 参数对象
- callback: Function 回调函数

- return: Object

参数对象

```
{
      body: 'JSAPI支付测试',
    total_fee: 1,
    trade_type: 'JSAPI',
    openid: 'oUpF8uMuAJO_M2pxb1Q9zNjWeS6o',
    serverId: '',
    playerId: '',
    userId: ''
}
```

返回对象

```
{
    return_code: 'SUCCESS',
    return_msg: 'OK',
    appid: 'wx2421b1c4370ec43b',
    mch_id: '10000100',
    nonce_str: 'IITRi8Iabbblz1Jc',
    sign: '7921E432F65EB8ED0CE9755F0E86D72F',
    result_code: 'SUCCESS',
    prepay_id: 'wx201411101639507cbf6ffd8b0779950874',
    trade_type: JSAPI
}
```

微信支付统一下单方法, 在使用微信jssdk支付时, 需要先统一下单, 才能返回需要的prepay_id

sgt.WxCentralService.auth(appid, scope)

- appid: string 微信的appid
- scope: string 可选 应用授权作用域，snsapi_base （不弹出授权页面，直接跳转，只能获取用户openid），snsapi_userinfo （弹出授权页面，可通过openid拿到昵称、性别、所在地。并且，即使在未关注的情况下，只要用户授权，也能获取其信息）

微信授权方法,使用此方法后,会在刷新当前页面,并在自动在SgtApi.context中注入参数openid的值, 如果scope为snsapi_userinfo时, 则会额外注入参数access_token的值
之后可以通过sgt.context.openid或者sgt.context.access_token去获取值, 另外,在localStorage中也保存了这两个值,格式为'sgt-' + SgtApi.context.appId + '-openid'或'sgt-' + SgtApi.context.appId + '-access_token'

sgt.WxCentralService.getUserAccessToken(code, callback)

- code: string auth验证返回的code
- callback: Function 回调函数
- return: Object

```
{
   "access_token":"ACCESS_TOKEN",
   "expires_in":7200,
   "refresh_token":"REFRESH_TOKEN",
   "openid":"OPENID",
   "scope":"SCOPE",
   "unionid": "o6_bmasdasdsad6_2sgVt7hMZOPfL"
}
```

此方法一般不用调用, 通过auth方法后会自动执行此方法(通过code换取网页授权access_token还有openid)

sgt.WxCentralService.getUserInfo(callback)

- callback: Function 回调函数
- return: Object

```
{
   "openid":" OPENID",
   " nickname": NICKNAME,
   "sex":"1",
   "province":"PROVINCE"
   "city":"CITY",
   "country":"COUNTRY",
    "headimgurl":    "http://wx.qlogo.cn/mmopen/g3MonUZtNHkdmzicIlibx6iaFqAc56vxLSUfpb6n5
    "privilege":[
    "PRIVILEGE1"
    "PRIVILEGE2"
    ],
    "unionid": "o6_bmasdasdsad6_2sgVt7hMZOPfL"
}
```

登录获取微信用户信息,需要在auth授权scope参数是snsapi_userinfo时,授权之后才能使用此方法, 否则此方法必然会失败

文件分发存储 - FileStorageService

文件分发存储业务 此业务不提供创建/上传文件接口， 如有需要请在客户端中访问客户端sdk提供的接口

sgt.FileStorageService.getUrl(fileName, callback)

- fileName: string 文件路径/key值
- callback: Function 回调函数
- return: string

获取指定文件路径可访问的url

sgt.FileStorageService.delete(fileName, callback)

- fileName: string 文件路径/key值
- callback: Function 回调函数
- return: boolean 删除成功true,否则返回false

删除文件

用户留资 - UserLeaveInfoService

留资模块
提供用户留存资料接口

sgt.UserLeaveInfoService.saveLeaveInfo(userLeaveInfo, callback)

- userLeaveInfo:UserLeaveInfo 用户留资对象
- callback:Function 回调函数
- return:UserLeaveInfo 用户留资对象 保存用户留资方法

Class:UserLeaveInfo

- id 主键id
- userName 用户名
- phone 联系电话
- address 联系地址
- content 备注
- createTime 创建时间
- updateTime 更新时间
- appId 所属appid

随机角色名 - RandomNameGroupService

随机角色名生成模块
提供了生成随机角色名接口，可以使用默认的姓和名文本库生成角色名，还可以通过开发者管理后台自定义上传姓和名文本库生成角色名。

sgt.RandomNameGroupService.defaultRandomName(callback)

- callback:Function 回调函数
- return:string 角色名 从默认的文本内容 生成随机名字

sgt.RandomNameGroupService.randomNameByGroupName(groupName, callback)

- groupName:string 库名（需通过开发者管理后台--数据管理--随机名称页面新增一条自定义文本库信息）
- callback:Function 回调函数
- return:string 角色名 根据groupName 指定的库文本内容 生成随机名字

Socket服务 - SocketService

使用此服务需要在页面另外导入socketio包

```
<script src="https://cdn.socket.io/socket.io-1.3.7.js"></script>
```

sgt.SocketService.getSocket(nameSpace)

- nameSpace: String (可选) 名称空间
- return: SocketIO SocketIO对象

此方法返回一个socketio实例对象, 请使用此socketio实例对象进行交互
例如

```
//获取socketio对象
var socket = sgt.SocketService.getSocket();

//监听连接事件
socket.on('connect', function(){
    //发送消息事件
    socket.emit('mass');
})
```

需要查看socketio的教程请访问http://socket.io/
需要查看利用sgtcloud-html5-sdk做出的demo可以访问demo-lobby-room

目前已内置事件

- mass 群发消息，整个命名空间的客户端都会收到
- message
- roomMass 房间消息, 整个房间的客户端都会收到
- gameBegin
- gameOver
- createRoom 创建房间
- joinRoom 加入房间
- leaveRoom 离开房间