

sgtcloud-html5-sdk用户指南

www.sgtcloud.cn

Published
with GitBook



目錄

1. [介紹](#)
2. [概述](#)
3. [准备](#)
 - i. [技术架构](#)
 - ii. [重要概念](#)
 - iii. [如何构建](#)
4. [使用流程](#)
5. [功能介绍](#)
 - i. [角色 - PlayerService](#)
 - ii. [公告 - AnnouncementService](#)
 - iii. [签到 - CheckinBoardService](#)
 - iv. [排行榜 - LeaderBoardService](#)
 - v. [邮件 - MailService](#)
 - vi. [活动 - CampaignService](#)
 - vii. [Boss - BossService](#)
 - viii. [抽奖 - GachaBoxService](#)
 - ix. [黑名单 - BlackListService](#)
 - x. [反馈 - TicketService](#)

sgtcloud-html5-sdk

介绍

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

致力于发展为 mbass的, 开箱即用的, 超酷的 html5 软件开发工具.

了解更多, 请访问我们的主页 <http://www.sgtcloud.cn>

安装

- 直接下载最新版本的 release
- 解压该 release
- 在 \$release/dist 目录中选择你需要的sdk
- 引入sdk到你的项目中

入门

将 sdk 引入到项目中：

```
<script src="sgtcloud-html5-sdk.2.0.0.min.js"></script>
```

或则使用CDN的方式：

```
<script src="http://www.sgtcloud.cn/dist/sgtcloud-html5-sdk.2.0.0.min.js"></script>
```

点击 [申请应用标识](#)

首次使用需要初始化应用标识：

```
SgtApi.init({  
  appId: 'xxx'    //填写应用标识  
});
```

生成一个简单实体：

```
var user = new SgtApi.entity.User();  
user.userName = 'xxx';  
user.password = 'xxx';
```

再调用一个简单业务, 每个业务的最后一个参数都是一个回调函数：

```
SgtApi.UserService.register(user, function(result, data) {
    if(result) { //表示注册成功
        /* 返回用户注册信息 */
        //data
    }
});
```

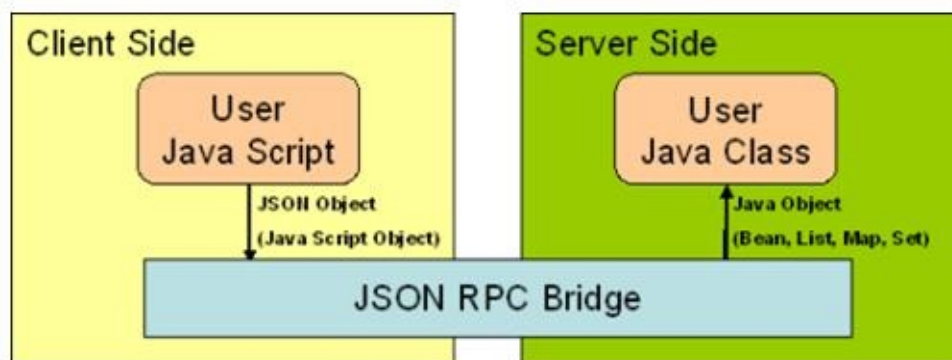
当登录/注册成功后, 可以操作该用户去执行其他业务

文档

- [官网](#)
- [API](#)
- [用户指南](#)
- [排行榜教程](#)

技术架构

sdk 采用了 [jsonrpc2.0](#) 的通讯协议



业务模块

- 用户
User - UserService
- 角色
Player - PlayerService
- 角色扩展
PlayerExtra - PlayerExtraService
- 成就
Achievement - AchievementService
- 公告
Announcement - AnnouncementService
- 活动

Campaign - CampaignService

- 签到

CheckinBoard - CheckinBoardService

- 任务

Task - TaskService

- 日常任务

DailyTask - DailyTaskService

- 好友

Friendship - FriendshipService

- 好友扩展

FriendshipExtra - FriendshipExtraService

- 黑名单

Blacklist - BlackListService

- 抽奖

GachaBox - GachaBoxService

- 排行榜

LeaderBoard - LeaderBoardService

- 邮件

Mail - MailService

- 通知

Notification - NotificationService

- 充值

Purchase - PurchaseService

- 商城

Store - StoreService

- 计费点

ChargePoint - ChargePointService

- Boss

Boss - BossService

- 文件存储

FileStorage - FileStorageService

- 礼包

GiftCode - GiftCodeService

- 个人通道
PrivateChannel - PrivateChannelService
- 公共通道
PublicChannel - PublicChannelService
- 第三方
DelegateDid - DelegateDidService
- 结构化数据
StructuredDate - StructuredDateService
- 反馈
Ticket - TicketService

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

License

The MIT License(<http://opensource.org/licenses/MIT>) 请自由地享受和参与开源

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。

准备

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.2.0.0.min.js"></script>
```

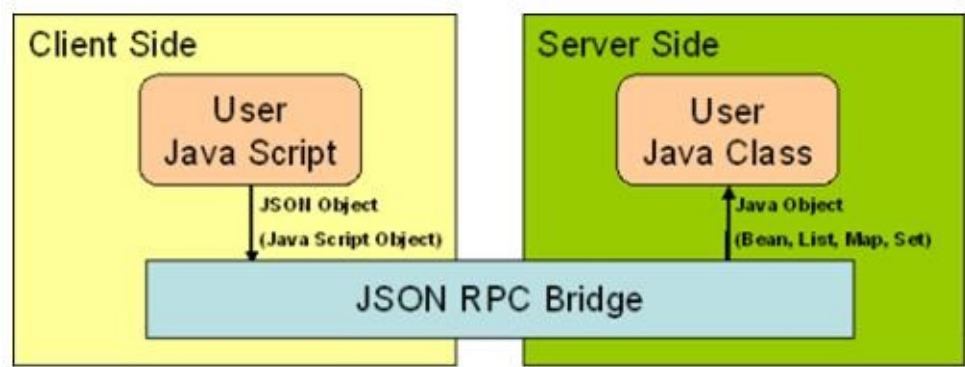

技术架构

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

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RouterService，UserService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，详细的方法请参看api.

各类 **Entity**

例如User，Server等Entity实体类抽象了sgtcloud平台的业务资源.
开发者也可以根据需求自定义一些业务实体，详细的方法请参看api.

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.

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
```

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**

初始化

设置应用标识

```
SgtApi.init({
  appId : 'html5_demo2015',    //应用标识
});
```

初始化成功之后,就可以使用 SgtApi 或者 \$s 的各种接口了

创建实例

因为js没有用到单例, 构建实例部分主要侧重于两种方式构造对象

普通方法创建实例

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

工厂方法创建实例

```
var user = SgtApi.entityFactory('User');
user.userName = 'xxx';
user.nickName = 'xxx';
user.password = 'xxx';
```

创建服务

注册User

```
SgtApi.UserService.register(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

```
SgtApi.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

```
SgtApi.context = {  
    userData: {},    //用户信息  
    serverData: {}   //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中内置了各种接口

下面我们将介绍核心的接口以及服务

角色 - PlayerService

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。

另外，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。

创建角色

```
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
    }
});
```

获取角色

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

更新角色

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

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

删除角色

```
SgtApi.PlayerService.deletePlayerByPlayerId(playerId, callback);
```

下载存档

```
SgtApi.PlayerService.downloadSave(playerId, callback);
```

根据最后登陆时间查找角色

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

根据角色名查找角色

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

根据用户ID查找角色

```
SgtApi.PlayerService.getByUserId(userId, callback);
```

获取指定角色的好友上限

```
SgtApi.PlayerService.getFriendsMaxNumber(playerId, callback);
```

通过用户ID查找其中的一个角色

```
SgtApi.PlayerService.getOneByUserId(userId, callback);
```

通过自定义ID获取角色信息

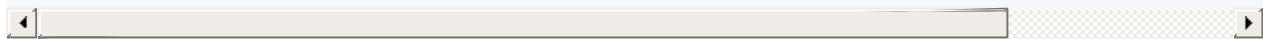
```
SgtApi.PlayerService.getPlayerByCustomId(customId, callback);
```

随机返回若干个最近登录的player

```
SgtApi.PlayerService.searchPlayerByLastLogin(limit, callback);
```

根据条件过滤并随机查询若干个最近登录的**player**

```
SgtApi.PlayerService.searchPlayersByLastLoginCondition(lastLoginTime, limit, excludePlaye
```



设置指定角色的好友上限

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

上传存档

```
SgtApi.PlayerService.uploadSave(save, callback);
```

公告 - AnnouncementService

公告业务接口

通过公告类型获取最新公告

```
SgtApi.AnnouncementService.getAnnounceByType(type, function(result,data) {  
    if (result) {  
        data;  
    }  
});
```

签到

签到主要分两种类型，一种是每日签到，即每日只能签到一次，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。

签到

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

获取最大累计签到数

```
SgtApi.CheckinBoardService.accumulateCount(playerId, checkinBoardId, callback);
```

获取连续签到数

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

通过CheckInBoardId获取签到板实体数据

```
SgtApi.CheckinBoardService.getCheckinboardByChekinboardId(checkinBoardId, callback);
```

获取最后签到时间

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

获取奖励

```
SgtApi.CheckinBoardService.getRewardByCheckinBoardId(checkinBoardId, callback);
```

补签

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

判断是否可以签到

```
SgtApi.CheckinBoardService.validateCheckin(playerId, checkinBoardId);
```

获取当前可用（有效期内）的签到板

```
SgtApi.CheckinBoardService.getAvailableCheckinBoards(callback);
```

根据自定义标签获取当前可用（有效期内）的签到板

```
SgtApi.CheckinBoardService.getAvailableCheckinBoardsByTag(tag, callback);
```

根据类型获取当前可用（有效期内）的签到板

```
SgtApi.CheckinBoardService.getAvailableCheckinBoardsByType(type, callback);
```

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）。

更新分数值

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

通过排行榜的leaderBoardID获取leaderBoard信息

```
SgtApi.LeaderBoardService.getLeaderBoardByLeaderId(leaderBoardId, 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.getLeaderBoardScoreByExample(leaderBoardId, player, callback);
```

通过排行榜ID和角色ID获取该角色的排行榜


```
SgtApi.LeaderBoardService.getLeaderBoardScoreByLeaderIdAndPlayerId(leaderBoardId, playerId,
```

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.getLeaderBoardScoresByLeaderIdAndPlayerId(leaderBoardId, playerId,
```


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.getTopLeaderBoardScoreByLeaderId(leaderBoardId, start, limit, c
```



提交排行榜数值

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



邮件

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。

发送一封邮件

```
SgtApi.MailService.sendMail(mail, callback);
```

接收邮件

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

接收未读取的邮件

```
SgtApi.MailService.receiveUnread(playerId, callback);
```

阅读邮件/批量阅读邮件

```
SgtApi.MailService.readMail(mailId, callback);
```

阅读一封邮件并领取附件

```
SgtApi.MailService.readAndPickAttachment(mailId, callback);
```

删除封邮件/批量删除邮件

```
SgtApi.MailService.deleteMail(mailId, callback);
```

获取所有未读和已读的邮件集合

```
SgtApi.MailService.getReadedAndUnreadedMails(playerId, callback);
```

领取邮件附件

```
SgtApi.MailService.pickAttachment(mailId, callback);
```


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。

获取当前已经激活的活动

```
SgtApi.CampaignService.getAvailableCampaigns(callback);
```

通过时间区间获取活动

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

通过活动ID获取活动

```
SgtApi.CampaignService.getCampaignById(campaignId, callback);
```

通过活动ID获取活动详情数据

```
SgtApi.CampaignService.getCampaignDataById(campaignId, callback);
```

通过活动详情ID获取活动详情数据

```
SgtApi.CampaignService.getCampaignDetailById(campaignId, callback);
```

获取进度

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

更新进度

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


Boss

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

通过id字符串获取boss数组

```
SgtApi.BossService.getByBossIdstr(ids, callback);
```

批量获取boss数据

```
SgtApi.getByBossIdint(id, callback);
```

通过bossId获取Boss实体

```
SgtApi.getByBossId(id, callback);
```

更新boss血量

```
SgtApi.attack(boosId, damage, callback);
```

获取boss当前血量

```
SgtApi.getCurrentHP(bossId, callback);
```

获取最后击杀人

```
SgtApi.getLastAttackPlayer(bossId, callback);
```

抽奖

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

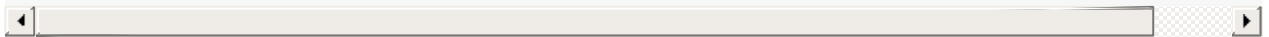
首先需要获得GachaBox,可以获得当前所有有效的GachaBox，也可以通过名称获得指定的GachaBox。

有自动修正的连抽

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

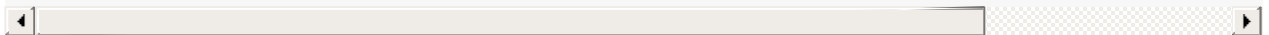
指定初始品质的自动修正连抽

```
SgtApi.GachaBoxService.autobalanceDrawQuality(playerId, gachaBoxId, num, quality, callbac
```



指定初始品质和最大品质的自动修正连抽

```
SgtApi.GachaBoxService.autobalanceDrawMaxQuality(playerId, gachaBoxId, num, quality, maxQ
```



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.draw(playerId, gachaBoxId, quality, callback);
```

获取当前所有有效的**GachaBox**

```
SgtApi.GachaBoxService.getAvailableGachaBox(callback);
```

获取指定名称的**GachaBox**

```
SgtApi.GachaBoxService.getGachaBoxByName(gachaBoxName, callback);
```

获取指定**gachaBox**的奖品

```
SgtApi.GachaBoxService.getLotteriesByGachaBoxId(gachaBoxId, callback);
```

指定奖品品质总值的连抽

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

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。

查询黑名单

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

更新黑名单

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

反馈

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

4.10.2. 查询反馈

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。

通过反馈者playerId获取自己发起的反馈信息

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

提交反馈

```
SgtApi.TicketService.sendTicket(ticket, callback);
```