Wallspace 前端缓存实现机制

#theCachePage和#thelist分别指代id为theCachePage和thelist的<ul>标签，

其中#thelist为数据最终展示区，#theCachePage为缓存数据暂存展示区

1. 底层机制是使用html5提供的接口localstorage存储，并对localstorage进行简单的封装。
2. 页面加载初始化global.initDataInPage = 1; 为了加载第一页的缓存数据。
3. Js调用callSDK("getJID");返回的global.JSFN.getJIDCallback() 中调用global.loadSpaceData();加载缓存数据并检查更新状态。
4. global.loadCache() 定义全局变量和加载缓存
   1. 定义全局变量

global.hasReachBottomCacheKey //保存刷新是否到底状态的key

global.curScanPageNo //存储当前要加载的缓存的页码

global.firstPageContCacheKey //第一页页面内容的key

global.PageContCacheKey //当前页页面内容的key

global.totlePageNumCacheKey //总共缓存的页数key

global.totlePageNum //从localstorage中读取的总缓存页数值，没有则置为0

global.spacePullUpLastModKey 下次数据加载的标记时间key

global.spaceUpdatedTS //圈子最新操作时间key

* 1. 加载缓存

只有在存在缓存并且是第一次初始化页面(下拉刷新不加载)或者上拉刷新时才加载缓存。

* + 1. 如果符合加载缓存的条件
       1. global.initDataInPage = 0; //标记下次加载缓存该变量已经不成立。只能通过上拉刷新条件进入。
       2. 将数据暂存在#theCachePage的<ul>中，在展示上给用户数据已经加载的视觉，在内容上是为了后期的其他操作做判断。
       3. global.curScanPageNo++; 将要加载的缓存游标变量下移
       4. global.hasLoadCache = true; //标记已经加入缓存
    2. 如果不符合加载缓存条件

global.hasLoadCache = false; //标记是否加载缓存的全局变量

1. global.checkIfUpdated();检查圈子动态是否更新
   1. 从localstorage中读取global.spaceUpdatedTS(圈子最新操作时间)，不存在则将变量置为0(无缓存时)
   2. 判断是否存在global.curPullType,不存在则将其置为1(默认为下拉刷新) global.curPullType = 1;
   3. 造参数调用SDK，检查更新

callSDK("ifUpdated",param);

1. global.JSFN.checkIfUpdatedCallback(data\_cbk) 检查更新

根据更新状态、缓存状态和刷新状态进行不同操作

* 1. 服务器端返回错误信息时，显示错误信息，将暂存在#theCachePage的数据追加在#thelist中并直接返回。
  2. 数据有更新时
     1. global.curPullType = 1;//刷新状态置为下拉刷新
     2. global.clearCache();清空缓存，准备接受新数据
     3. global.spacePullUpLastModKey对应数据置为较大值6212592000000
     4. global.spaceUpdatedT 对应数据置为圈子最新操作时间
     5. 调用global.newRefreshPage();获取新数据
  3. 数据无更新时

检查是否有缓存以及刷新状态

* + 1. 无缓存且上拉刷新时

global.newRefreshPage(); 获取新数据

* + 1. 无缓存且下拉刷新时

无操作或者显示提示信息

* + 1. 有缓存时

判断刷新类型

* + - 1. 下拉刷新时，将#theCachePage的数据追加到#thelist中，并清空#theCachePage。
      2. 上拉刷新时

获取#theCachePage中的<li>的数量

* + - * 1. 当数量为20(指定的每页数量)时，将#theCachePage的数据追加到#thelist中，并清空#theCachePage。
        2. 当数量为0时，判定缓存数据为空数据(一般执行不到)，global.newRefreshPage(); 获取新数据。
        3. 当数据小于20时，与等于20时一样。

1. global.newRefreshPage数据刷新或获取新数据操作

从缓存中取得 window.spacePullUpLastModKey对应的数值，不存在则置为6212592000000。

* 1. 构造参数并调用SDK callSDK("getPostListByTS",param)，获取新的post列表。

1. global.JSFN.postRefreshCallback 动态列表刷新

检查是否有数据，并根据#thelist和刷新状态给出不同的提示或者继续操作

* 1. 将json数据取出并使用模板生成DOM结构
  2. 更新缓存

//更新总缓存页数

ws.setTotalCachePage(global.totlePageNumCacheKey,totle)

ws.setData(dataKey,html);//设置当前页的缓存数据

ws.setLastModify(global.spacePullUpLastModKey,data\_cbk.lastModify);//设置下次获取新数据的标记时间节点

* 1. 将数据追加到#thelist中，#theCachePage置空并刷新页面。

完成一次页面的展示和缓存。

缓存机制流程图

