Lizard2.1最佳实践

# 频道内页面切换控制

Lizard处理频道内跳转的处理步骤为

* 根据url向服务端发get请求，拿到对应动态页面编译后的HTML字符串
* 解析字符串中的lizard-config配置，请求数据、渲染模板
* 根据lizard-config中的controller配置，拿到对应视图的构造函数，根据上一步渲染好的视图节点完成View的实例化
* 显示新的视图

实际业务场景可能有如下场景

1. 在完成模板渲染后，其实可以先把Dom节点显示出来，尤其是页面没有首屏数据请求的时候。此时框架的显示时机有点偏晚。
2. 基本是上面的步骤全部做完，业务仍会发一个Ajax请求，完成View中某些区域的渲染，此时业务才需要显示View。这个时候框架的显示时机又有点偏早。

针对如上两个问题的解决方案

1. Lizard-config扩展showfake选项和controller配置平行，如下

"controller": "/webapp/car/webresource/chf/index.js",

"showfake": {hideloading: true}

框架会识别这个showfake如果配置了，则会在节点渲染完毕后直接显示出来，其中子配置项用于控制在新的 View节点显示出来后是否隐藏loading，这个时候由于仅仅是完成了Dom节点的构造，但是View的实例化尚未完成，事件尚未处理，所以建议各BU还是继续显示loading狂

1. 针对这个问题，用户在定义自己的View的时候设置switchByOut属性为true，代码片段如下

define(['cPageView'........................................],

function (BasePageView................................) { var View = BasePageView.extend({

.........

switchByOut: true,

对于设置了这标志为的View框架会识别，即便是完成了View的实例化后也不会执行这个View的show以及上一个View的hide，这个时候需要业务调用turning方法，完成View的显示。

# 频道首页的性能调优

从浏览器输入地址到频道首页的显示，其中的步骤是

* 请求页面document文档，下载JS,CSS等资源
* 解析页面中的lizard-config配置，后面的步骤同频道见跳转

我们看到，如果首页要想先出来，则需要等页面加载完框架JS以及CSS并等框架完成页面的解析、渲染。如果网络为2.5G网络，则百页的时间会比较长，而且首页往往页面元素也比较简单，一个简单的页面，留给用户太多的白页时间，体验是在不好。

解决方案：

首页的DOM往往比较固定，因此首页的DOM元素可以直接现在首页的页面中，这样只要文档和CSS加载完毕，首页就可以显示在用户眼前了。在首页的实例化完成后，在首页View的onCreate方法中可以删除首页的“假”节点。

为了更近一步，可以对这个假页面的几个关键节点绑定事件，这些事件被处罚后，显示是一个假的loading浮层（这些DOM节点也是固定写在首页中的），并用全局变量记录点击的节点。

在View的onShow中根据全局变量是否存在，来判断是否出发真正的View中节点的事件。

# 频道内页面显示效率调优

作为解释型语言，JS和编译语言最大的不同就是编译语言在编译时就需要引入依赖的模块，而JS可以在运行时引入。

每一个页面的View显示出来后，有些交互操作会引发UI 组件的实例化操作，比如点击后弹出浮层选择时间。目前的实现时业务在定义View的时候，把这个UI组件直接作为View的依赖项了，那么requireJS在加载View模块的时候会加载这个UI组件。实际的情况是，这个View用户可能不会点开这个组件，针对这种情况，框架对View提出了迟滞加载的概念对于首屏显示不需要的组件，允许业务定义在组件的一个属性中，框架会在JS执行的空闲时间片中去加载这些模块。提升性能的同时，榨干空闲时间片的使用效率。

样例代码如下：

define(['cPageView'........................................],

function (BasePageView................................) { var View = BasePageView.extend({

........

defferModules: ['UIBubbleLayer', 'UITab'],

defferModuleCallback: function(UIBubbleLayer, UITab){

this. UIBubbleLayer = UIBubbleLayer;

this. UITab = UITab;

},,

其中defferModules属性可以用来定义首屏不需要，但在交互过程中可能用到的组件模块，defferModuleCallback则是加载完这些模块后的回调。

# 多webView跳转封装

在Lizard.goTo(forward)（也对应于View的forward函数）函数中，框架会识别当前环境的UA,如果在携程旅行APP内，则会识别targetModel，实现多webView以及往native页的跳转，参见

http://svn.ui.sh.ctripcorp.com/lizard/2.1/doc/Global.Lizard.html



# 头部数据格式统一

Lizard2.1废弃了 customtitle属性，统一了H5和Hybrid下的格式设置。

title/subtitle设置，参见

<http://svn.ui.sh.ctripcorp.com/lizard/webapp/demo2.1/index.html#%2Fwebapp%2Fdemo2.1%2Fui%2Fnew_header>

侧边栏设置，参见：

<http://svn.ui.sh.ctripcorp.com/lizard/webapp/demo2.1/index.html#%2Fwebapp%2Fdemo2.1%2Fui%2Fheader_sidebar>

头部城市列表选择设置

<http://svn.ui.sh.ctripcorp.com/lizard/webapp/demo2.1/index.html#%2Fwebapp%2Fdemo2.1%2Fui%2Fheader_select>

头部中间的按钮设置：

<http://svn.ui.sh.ctripcorp.com/lizard/webapp/demo2.1/index.html#%2Fwebapp%2Fdemo2.1%2Fui%2Fheader_tabs>

# 对于已经存在View的节点处理

Lizard2.1View切换的处理机制为首先解析lizard-config看下对应的 ctrollerJS对应的 View是否已经实例化过，如果实例化过，则不再实例化。对于已经实例化过后的View则进一步判断当前URL和之前这个 View的URL 是否一致，如果一致则直接显示，否则会继续判断这个View是否定义了emptyContentNode方法，如果定义了，且返回为true则会重新渲染View的节点，否则直接显示。

# PUB/SUB机制

Pub/Sub 模型定义了如何向一个内容节点发布和订阅消息，这些节点被称作主题(topic)。

Lizard2.1定义了cMessageCenter模块，用以实现主题的发布订阅，这个机制可以被用来实现View之间的通信。具体的实现方式

订阅主题的代码

define(['cMessageCenter'], function(cMessageCenter){  
  cMessageCenter.subscribe('message', function(param1,param2){  
       
  })  
})

发布主题的代码  
define(['cMessageCenter'], function(cMessageCenter){  
  cMessageCenter.publish('message', [param1,param2])  
})