全平台、多账号体系的58微聊前端系统架构实践

摘要

58微聊前端系统，是一套支持全平台、多账号体系的前端解决方案。目前已经接入房产、招聘、二手车、黄页等各大业务线，潜移默化中推动着58方方面面蓬勃发展。

简介

沟通是人类赖以生存的基础。在58同城这个中国最大的分类信息生活服务平台中，58微聊承担着沟通的纽带的作用。

58微聊承载着服务用户，连接团队，助力未来的艰巨使命；目前已经接入房产、招聘、二手车、黄页等各大业务线，潜移默化中推动着58方方面面蓬勃发展，尤其是在服务用户方面很好的契合了我们的使命：“通过互联网让人们生活简单美好”。

58微聊作为即时通讯工具，提供Android、iOS、PC、Web、Wap、小程序全平台的支持。下面将具体分析58微聊的前端技术架构。

整体架构

上图是微聊前端系统的整体架构，主要分为网络、SDK、UI、接入四层及存储、安全、质量监控、polyfill等模块。

各层及模块的功能如下：

- 网络层：提供统一的短连接、长连接接口调用。
- SDK层：提供统一的业务接口封装，以及核心的业务逻辑管理。
- UI层：UI层是体现平台差异最大的地方，采用了完全不同的设计风格，所以各端的UI实现完全独立实现。
  PC版需要兼容IE8-，采用原生语法模块化开发；M端则使用VUE框架；小程序端由于目前相关框架问题还比较多，采用原生小程序语法开发。
- 接入层：58集团旗下拥有58、赶集、安居客等独立运营的业务线，他们各自有自己的账号体系及PC、M、小程序的产品，所以UI上很难做到适合所有的业务需求。
  针对各端我们开发了一款默认的UI，各业务线可以基于默认的UI进行定制，也可以只接入SDK，自行实现UI界面。
- 存储模块：为了使用的便捷和最大限度地复用代码，存储模块屏蔽了各平台的本地存储方案差异，提供了统一的存取接口。
- 安全模块：通过对请求数据加密，防止低成本地进行接口模拟调用；通过对渲染的数据过滤，防止XSS攻击。
- polyfill模块：为了使用更多便捷、高效的接口并兼容各平台，将一系列的polyfill进行统一管理、按需加载
- 质量监控模块：上报用户场景下系统的一些关键指标，弥补在开发、测试环节发现不了的问题，为后续优化提供指导。

消息中心

消息中心是SDK层最核心的功能，简单的来说就是接收服务器端下发（通过websocket或者HTTP长连接）的消息。但不管是websocket还是HTTP长连接（下称通道），对服务器端及客户端的资源消耗都是非常大的。

在PC上同时打开多个页面是非常常见的事情，如果每个页面都建立一个通道，资源消耗将是非常巨大的，尤其是服务端开销。

为了解决以上问题，我们设计、封装了多页面消息同步的消息管理器，确保了同一个浏览器同时有且只有一个通道存在，所有的消息都通过这个唯一的通道下发。

这里边有两个核心问题需要解决：如何确保有且只有一个通道？通道下发的消息如何同步到其他页面?

第一个问题：我们在所有页面共享一个心跳（本地存储的一个字段），在存在通道的页面定时（如：3秒）更新心跳时间，在没有通道的页面定时（如：5秒）检测心跳是否有更新，如果有则说明有其他页面存在通道，否则马上建立新的通道（因为没有任何页面维持通道了）。

第二个问题：消息通过唯一的通道下发后，最直观的方式就是写入本地存储，其他页面定时去查询。但需要更新数据的地方非常多，如果每个模块都去定时读取数据，程序将会非常复杂并难以管理。我们的做法是，统一封装了一个消息管理器，提供统一的订阅/发布接口，其他需要数据的模块只需要统一的订阅消息管理器特定类型的消息就可以了。

消息管理器主要负责两个核心功能：接收通道消息并写入本地存储器和定时读取本地存储器数据并回调给已订阅的模块。

消息中心的流程如下：
消息管理器类图如下：

跨平台兼容

58微聊前端项目在58集团内的使用非常广泛。
从端来看，需要支持PC、M、和小程序。
从业务线来看，需要支持58、赶集、安居客等，他们都有各自的账号体系。
从浏览器兼容性来看，PC上需要兼容IE8及以下版本，M端需要兼容IE9+版本，小程序暂无太多需要兼容处理的地方。

一方面，多端、多业务线的支持给技术选型带来了很大的挑战，最简单直观的就是各端各自为政，进行独立开发、维护。
但由于业务逻辑的一致性，三端的业务代码会有非常高的重合度，
这意味着每一次的改动需要同时修改三个代码库，这既浪费时间，又难以做到修改同步。
基于此，我们把业务重合度高的SDK层进行合并、统一开发、管理，最大限度复用代码，降低维护成本。
而对区别非常大，复用性非常小的UI层则完全独立开发、迭代，有效避免技术选型的局限性。

另一方面，虽然ES6标准都已经发布很久了，但对于IE9-的浏览器连ES5都不支持，所以很多简洁的语法不能直接应用到项目中，
不过像`Function.prototype.bind`，`Function.prototype.filter`这类的语法糖，已经有很成熟的polyfill方案，
所以在项目中可以在引入这些polyfill的情况下，尽情享受这些新语法带来的便利。但兼容代码的引入势必会增大代码的体积，
58微聊项目的解决方案是，在业务代码编写的时候使用新的语法，但只在WebSdk中引入polyfill以实现对低版本浏览器的兼容，
这样做，既可以在所有代码中使用新的语法，也不会对不需要兼容的平台（M端和小程序）引入额外的代码，实现了代码利用的最优化。

质量监控

网络环境千变万化，用户的设备、系统、网络千差万别或者说千奇百怪的，不管是开发还是测试人员的环境都只是很小的一个子集。仅仅通过开发、测试人员很难覆盖所有场景，我们需要了解更多用户实际的情形，从而更全面地了解整个系统的质量并实施改进措施。

为了解决以上痛点，我们开发了前端质量监控组件，用于上报用户场景下系统的一些关键指标，从而收集在开发、测试环节发现不了的问题。

这些关键指标主要包括：

- 错误日志：可以有效监控代码BUG，在大量用户反馈之前拿到错误现场，可以更快地解决问题。也实实在在地为我们的系统发现了一些难以复现、只在特定用户环境出现的BUG，使得定位、解决问题的效率得到了本质的提升，而且还极大程度地降低了对用户的打扰，将问题扼杀在萌芽阶段，避免了更大规模的蔓延。
- 性能指标：可以有效监控代码质量，为我们后续的性能优化提供了基础和标杆，可以通过性能指标曲线方便、快捷地看到优化的效果。
- 劫持信息：可以有效把握网站安全现状，为后续的安全策略提供场景指导。后续我们会根据实际的劫持情况，开启CSP、RSI等安全策略。

写在最后

58微聊前端项目，作为58集团各服务平台的基础、核心功能，连接着平台上的各类用户，发挥着桥梁的作用。经过一两年的功能迭代，项目变得比较复杂，在追求卓越的58工程师文化的驱动下，我们不断地进行着代码的重构，从而达到目前的分层结构。

时代在变迁，技术在进步，任何系统或者产品都需要不断的迭代升级，以适应新的形势。58微聊前端项目作为集团核心的基础服务工具，我们也在不断地追逐更极致的用户体验。

短期来看，我们会充分利用质量监控收集的关键指标数据，针对性地进行代码质量和性能的提升。

中长期来看，我们会紧随技术变迁的步伐，探索、研究pwa、csp、webrtc等新技术运用到项目的可行性以及落地方案的制定和实施。