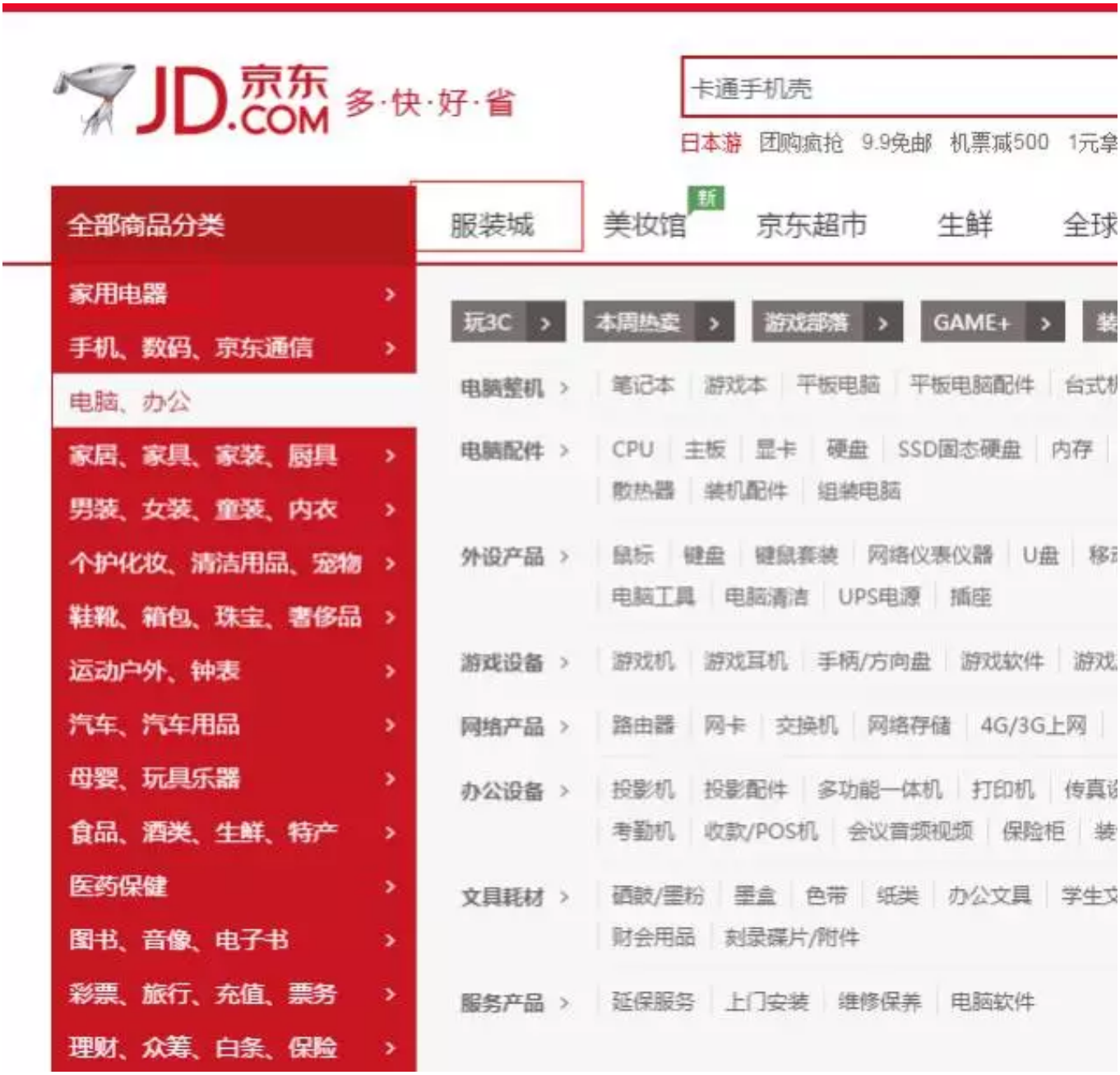
京东上千页面搭建基石——CMS前后端分离演进史

原创：于林坤 亿级流量网站架构 2016-07-29

作者：于林坤，2012年加入京东，网站移动研发部频道业务技术负责人，先后多次主导京东商城首页、频道页技改及架构升级，在高并发系统架构、前端系统架构与优化方面有丰富经验，PHPer。

京东CMS简介

CMS即内容管理系统（ContentManagementSystem），目的是用于快速进行网站建设或者网页开发。对于京东网站部门来说，CMS核心目的是用来快速开发和上线各种页面，诸如各种垂直频道页，访问www.jd.com将看到如下页面，如点击“服装城”、“家用电器”等都会跳转到一个垂直频道页；这些页面中有许多页面风格是类似的，因此很适合使用CMS进行快速搭建。



对于我们来说，CMS最核心的目的就是进行数据和模板的统一管理、页面的统一发布,从而减少之前的很多重复工作。

京东CMS是2014年提出来的，经过两年多的完善，目前已经发展为一个集标准服务管理、标准组件服务和智能投放于一体的标准化导购运营系统。**具有以下特点：**

1. 搭建快速，统一发布，统一架构；
2. 前后端分离，后端不再负责页面渲染，只提供高性能、可复用的API；
3. 移动端页面支持；
4. 数据分析、智能投放的特点；

业务支持场景：

首页、频道页、垂直页、活动页的搭建及单品页、列表页部分可维护的业务等。

从基本功能及架构来看，**可以分为三个阶段**：
CMS 1.0——虚拟分类系统
CMS 2.0——CMS系统
CMS 3.0——CMS-portal系统

CMS 1.0—虚拟分类系统

虚拟分类系统是一个独立的促销商品、促销文字维护系统，与具体前端业务架构脱离，哪条线接入虚拟分类，需要根据自己的业务逻辑单独开发、维护、部署自己的架构。说白了，虚拟分类系统提供一些基础数据；然后比如我要搭建一个家电频道页，则需要开发一个Web项目，然后调用虚拟分类系统获取数据然后进行模板渲染处理处理。因此虚拟分类系统当时只是一个基础数据维护平台，无法实现信息的共享、复用和集约化管理。这就会存在各种各样的频道页系统，导致管理混乱，性能上各有差异，出现过很多次事故。而且各系统需要独立配置，导致工作量大，占用资源多，无法快速响应业务需求。

下图是当时不同业务体系的架构：

可以看出部分频道页和活动页用的nginx+tomcat，部分频道及垂直站用的nginx+php-fpm，与虚拟分类联系不大，总体遵循各业务层获取虚拟分类的数据，分别独立上线、部署、维护，应用层直连mysql，mysql 抗不住，会增加一层 memcache。

CMS 2.0—CMS系统

Cms2.0总结了1.0时的不足，从节省资源、控制成本的角度考虑，把导购类的个体页面业务进行了统一，模板能复用的复用，以前虚拟分类系统的频道也需要迁移到新的系统。

我们做了以下改动：

1. CMS 2.0数据结构适合虚拟分类体系，方便新老数据兼容；
2. 升级架构，统一配置、发布流程；去memcache为redis，做大量性能压测来调优php-fpm配置，单机TPS能达到3000+； 配置定时任务，保证redis数据实时性；
3. 单点发布，一键预览增强采销维护数据的机动性；
4. 单机闭环，单机闭环服务设计是CMS整体架构的核心部分，需要展示的内容在本机获取、封装、校验；
5. 模块化、动态数据类型初期版本（CMS 3.0会细说）；

架构图：

对比1.0，新的CMS可以让频道页的开发周期降低2~4周，大大提高了业务需求的响应速度；它看起来更独立，更像一个整体，在容灾、规避风险方面做了严谨的优化；同时让采销在维护数据时，更方便、更简单。

后续由于个性化的需求越来越多，大量的频道业务需要开发人员一个一个套模板来实现，大大加大了开发人员的工作强度，之前的模板复用方式已经无法满足业务的需求，同时太简单的数据模块，需要手工来绑定数据类型也增加了开发成本，这时候需要我们必须做出改变。

CMS 3.0—CMS-portal系统

CMS 2.0后也存在很多痛点，因此我们也想在CMS3.0上**解决这些问题**：

1. 本质问题就是要快：快速支持、响应业务越来越多的垂直化页面改版；
2. 前后端分离，页面渲染让前端实现，解放后端让后端做高大上的事情；
3. 减轻运营人员的工作量，系统简单好用，提高导购类商品的转化率；
4. 其他系统的支撑，实现CMS的集约化管理；
5. 兼容手机端；
6. 站点管理、统一架构、容灾、高性能、水平扩容；

通过两版CMS系统的开发，我们发现CMS无外乎管理的是数据和模板，另外需要好的预览、一键发布支持。而传统数据管理都是静态数据类型，而我们做了动态数据类型的设计；另外我们做了模板管理中心，并抽象了模板、楼层、元件、模块的概念，从而实现更好的复用性。

统一架构

主要分为如下几部分

CMS系统：统一管理CMS相关数据，并进行页面的生成和发布；

数据Worker管理中心：调用第三方服务进行数据校验（如库存不足补货），并调用CMS系统进行页面生成和发布，发布到单点服务器上；

单点服务器：相关页面的单机闭环实现，即CMS发布的页面会存储在这些单点服务器上；每个机房会部署一台；

页面定时更新Worker：定期同步单点服务器内容到静态应用服务器集群，并保存历史版本，当出现问题时可以切换回上一个版本；

静态应用服务器集群：静态托底实现，会存储相关的静态页面文件，当单点服务器挂了时，降级到该集群；

异步加载的个性化服务：有些数据不能存储到静态页，如价格/库存/推荐等数据，此时通过异步加载这些数据，实现个性化服务；

接入层Nginx：接入层Nginx负责请求的路由和服务的降级。

主要思路

1. 引入动态数据类型；
2. 页面模板管理中心，模板、楼层、元件、模块设计，实现可复用；
3. 使用元件实现前后端分离；
4. 动态服务和业务数据闭环；
5. 预览、一键发布，单点管理；
6. H5版直接搭建，native版 API 支持；
7. 大数据智能选品应用；

动态数据类型

所谓的动态是指能灵活扩展的，不需要上线也不需要修改数据库字段，支持自由扩展；这样做的好处是能够快速响应电商网站的日益灵活多变的促销需求。比如促销语：

目前常用的数据类型为文字链、小图文、商品池等。

操作界面：

动态数据类型数据结构：

fields是json串，用于动态定义字段。

使用元件实现前后端分离

使用动态数据类型定义了数据之后，需要在模板中使用它。而在我们CMS系统中进行了页面、模板、楼层、元件、模块的划分。模块是某种数据类型的具体化，即有了数据的数据类型。元件是由模块和HTML代码段（根据模块数据进行渲染的一小段模板）组成；楼层通过一系列元件组成，而模板会引入多个楼层，当然也会引入一些JS、CSS等，最终通过模板渲染出相应页面。

type是数据类型表，module是模块表，source是数据表，按照上面的逻辑我们是通过数据类型获取到数据模块，并同时能拿到该模块所对应的商品数据（商品池）。

有了这个元件之后，就可以彻底解放后端，页面渲染工作完全交由前端来开发，实现了前后端的分离。

即CMS研发只负责平台和基础数据（动态服务）的维护，业务人员进行模块的维护，而前端人员独立完成元件开发、模板设计、开发和发布。

动态服务

跨线条业务间的资源复用、独立调用时需要提供相关的API，如三级地址服务，类目服务、动态加载的数据（如爆款特卖、今日推荐等）等。数据格式满足数据闭环原则。Lua+redis架构实现，单机（16U）QPS能达到20000+。

频道业务数据闭环

数据闭环，即数据的自我管理，或者说数据都在自己系统维护，不依赖与其他任何系统，去依赖化，这样得到的好处是别人抖动与我没关系。因此我们先要数据异构。

数据异构是数据闭环的第一步，将依赖系统的数据拿过来，按照自己的业务需求存储起来。频道业务需要异构的数据主要是三部分：商品基本信息、第三方数据、大数据。

数据原子化处理，数据异构的数据是原子化数据，这样未来我们可以对这些数据再加工再处理而响应变化的需求。我们有了一份原子化异构数据虽然方便处理新需求，但恰恰因为第一份数据是原子化的，那么它会很分散，前端读取时mget的话 性能不是很好，因此我们又做了数据聚合。

数据聚合，是将多个原子数据聚合为一个大JSON数据，这样前端展示只需要一次get，当然要考虑系统架构，比如我们使用的Redis改造，Redis又是单线程系统，我们需要部署更多的Redis来支持更高的并发，另外存储的值要尽可能的小。

容灾

应用层容灾

1. 数据校验，CMS在页面预览有一层严格的数据校验逻辑，比如数据格式、数据大小、敏感词等，保证页面生成100%没有问题。

2. 版本降级，静态页面出现问题，除了页面本身数据有问题外，潜入的js、css出现问题也会影响页面展示，这时候会版本降低为前一天的正确版本；

3. 异步服务，异步化数据容灾方面主要是监听服务的状态及响应时间；降级访问有隐藏该功能和切换服务器实现；

服务器容灾

主要是通过多机房部署，监控80端口，出现问题可以自动把流量水平切走。

智能选品

智能选品，是服务于前台的流量运营，为采销及运营人员提供运营支持，为每一次访问提供最合适和匹配的商品、品牌以及促销活动。是根据用户的行为推荐出相关的商品及活动。可以分为群体特征和个体特征。群体特征分为两部分，数据部分及规则部分。

数据部分是从大数据平台异构过来，当然这个数据是海量的，我们选择热点TOP 5000的数据来异构。

规则部分是通过京智后台创建，在审核通过后，触发MQ，通过ES 跑出对应数据，然后由频道页动态服务系统对外提供json格式的http服务。前端业务以异步的方式传递相关规则参数进行调用。

智能选品实现数据化、定制化、个性化、自动及半自动化内容运营。它可以模拟人脑选货逻辑，以运营指标为导向（GMV、订单转化率、点击量、毛利等），分区域、分用户选取最匹配的内容。目前应用于京东超市、行业频道以及618大促主会场，带来优于人工选品的转化效果，并解放采销运营人员日常繁琐的运营工作，提高了整体效率。

遇到的坑

rsync文件同步

上面的介绍过，我们的静态页面为了保持数据的一致性由单点服务器通过rsync同步静态文件到其他服务器，有时候会发现服务器负载无端的被打满。

分析问题发现如果定时任务脚本的同步未在规定时间内完成，crontab接下来的还会执行此脚本，这样就会产生相同的rsync的进程。按照这种状态，长时间就会衍生出很多个rsync进程，就会导致负载过高，甚至有些服务器会挂掉。这时候我们用到了rsync的进程锁，在目录下生成一个rsync.lock文件，当crontab执行时，rsync会判断锁文件是否存在，如果存在说明本次同步未完成，则不执行rsync。

数据一定要闭环

别人的接口抖动以及返回数据的异常，影响到前端展示对我们来说说是不能容忍的，这就需要我们针对各种情况一一校验。

CMS总结

目前通过CMS搭建、正在搭建以及使用CMS API支持的PC端、移动端页面约有上千个，这对CMS来说意义重大，随着业务需求量的越来越大，也给我们带来了新的期望。接下来，我们会从可视化编辑、数据统计分析、关键词管理、商品下架预警等方面进行相关的优化工作。

==========

喜欢我的内容请关注我的公众号。