H2H 技术、部署及架构

1：开发技术的选择：

1.1：开发语言的选择：php + html + css + JavaScript

后台为什么使用 php?

1：无论是在 windows server 上，还是 linux server 上，php 相比 java，更加容易部署。

2：java 的架构比 php 复杂得多，我们项目的特点：需求变化快，要求快速迭代，如果使用 java，难以应对快速变化的需求

3：对于响应速度的疑问：java 比 php 快？这个值得商榷，一般来说，web 应用的瓶颈在数据库，而不在于语言，退一万步讲，就算瓶颈在语言的选择，php 很多库都是 c 写的，而 java 类库都是使用 java，java 也未必就比 php 快；所以，我们选用 php 写后台。

1.2：数据库：mysql

1：因为开源且流行

1.3：服务器：apache

1：apache2 是目前的主流；

2：windows 上存在 wamp，linux 上存在 lamp，为了和 mysql、php 更好搭配，高效开发、快速迭代、快速部署，所以使用 apache2；

1.4：平台形式：网站

1：限于现有技术，我们难以做 app；

2：对于一个使用得不是很频繁的服务，没有必要让用户安装一个 app。

2：部署

项目部署在同一服务器下，不同身份的用户只是通过不同的域名访问

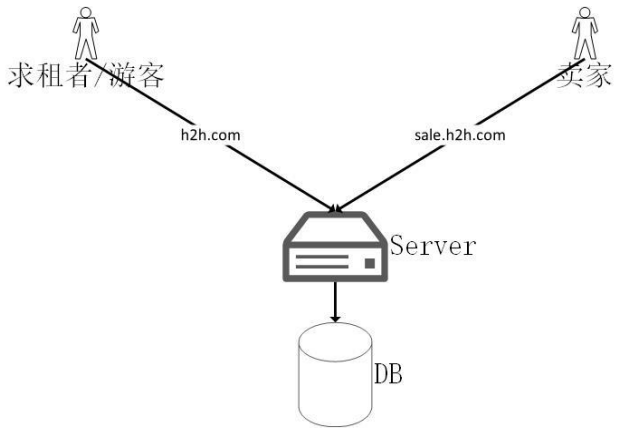

图 1：软件部署

3 软件架构

3.1:总体模块划分

H2H 整体架构基于三层架构，在三层架构的基础上进行了一些改进，具体模块划分如下图：

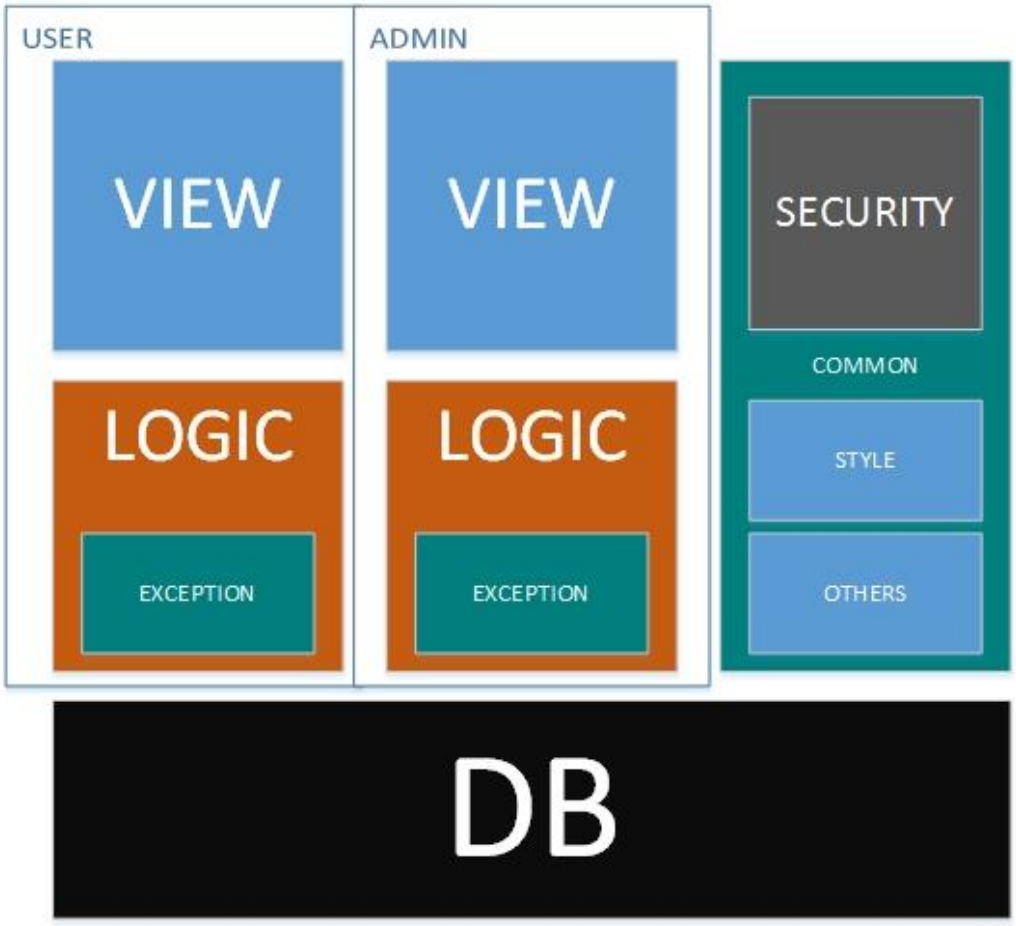

图 2：系统模块划分

这样划分主要出于以下几个方面的考虑：

1：可拓展性：首先，三层架构就已经实现了视图、逻辑、数据三个层的解耦，但这对于我们的系统来说，还是不够。虽然这是同一个工程，但它负责了用户与管理员两方。为了实现修改任何一方的代码都不影响另一方，我们决定将 user 与 admin 模块实现完全分离。

2：可重用：在我们的系统中，存在很多可重用的功能模块，比如系统 css 文件，安全性保障模块，我们将可重用的代码放到 common 模块中，提高代码重用性。

3：安全性：虽然 user 与 admin 在操作上互不干扰，但是系统后台对其响应确是修改同一个数据库，为了防止 user 与 admin 修改数据库的操作互相冲突，我们采用同一个数据库访问模块。

3.2:信息流

下面这张图展现了一个用户的服务请求是怎样被执行的：

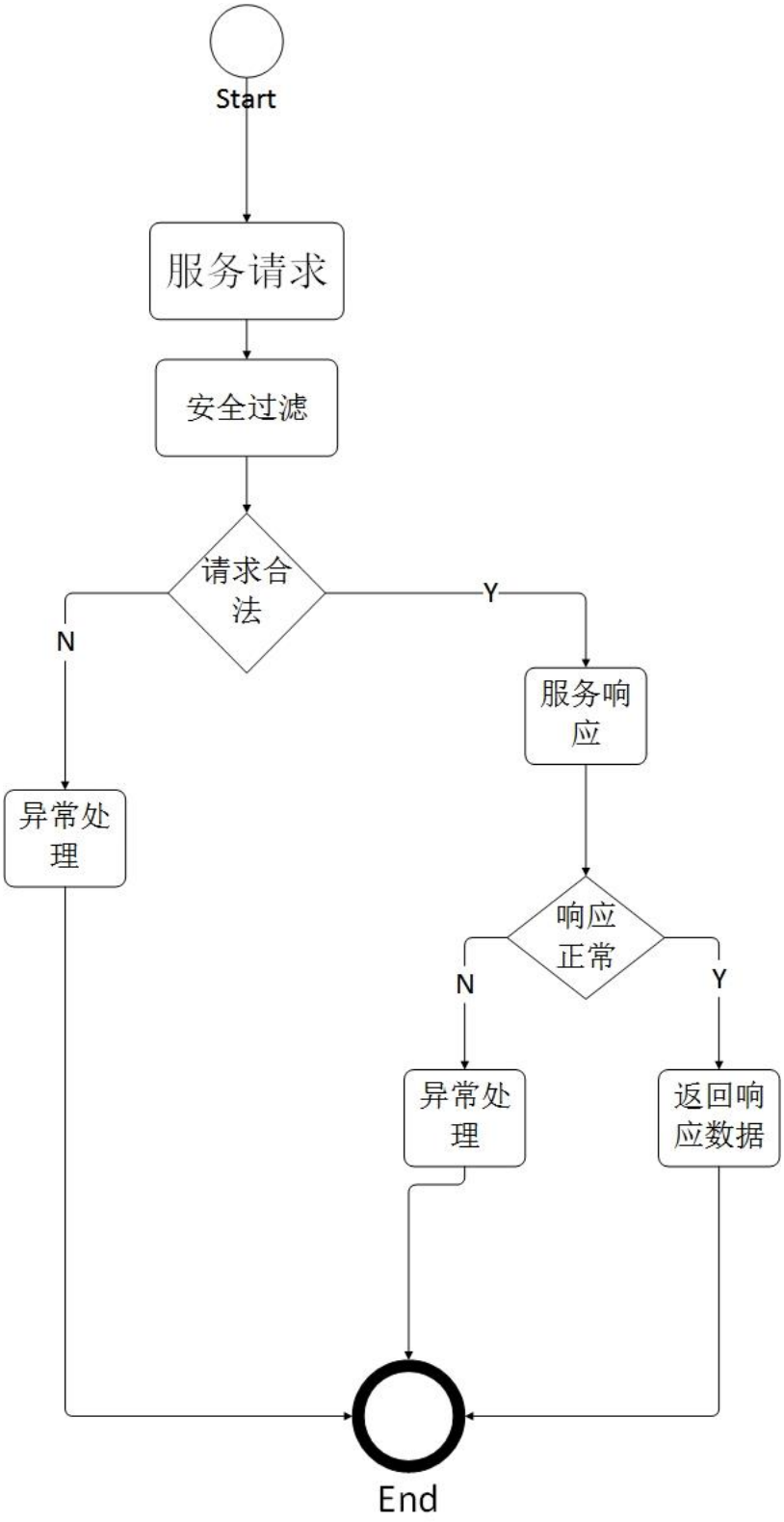

图 3：信息流

3.3:系统架构中的测试模块

与最后发行版本不同的是，在开发过程中，我们还在每个模块中添加了一个独立的小模块：测试模块。

所以，开发版本的架构如下：

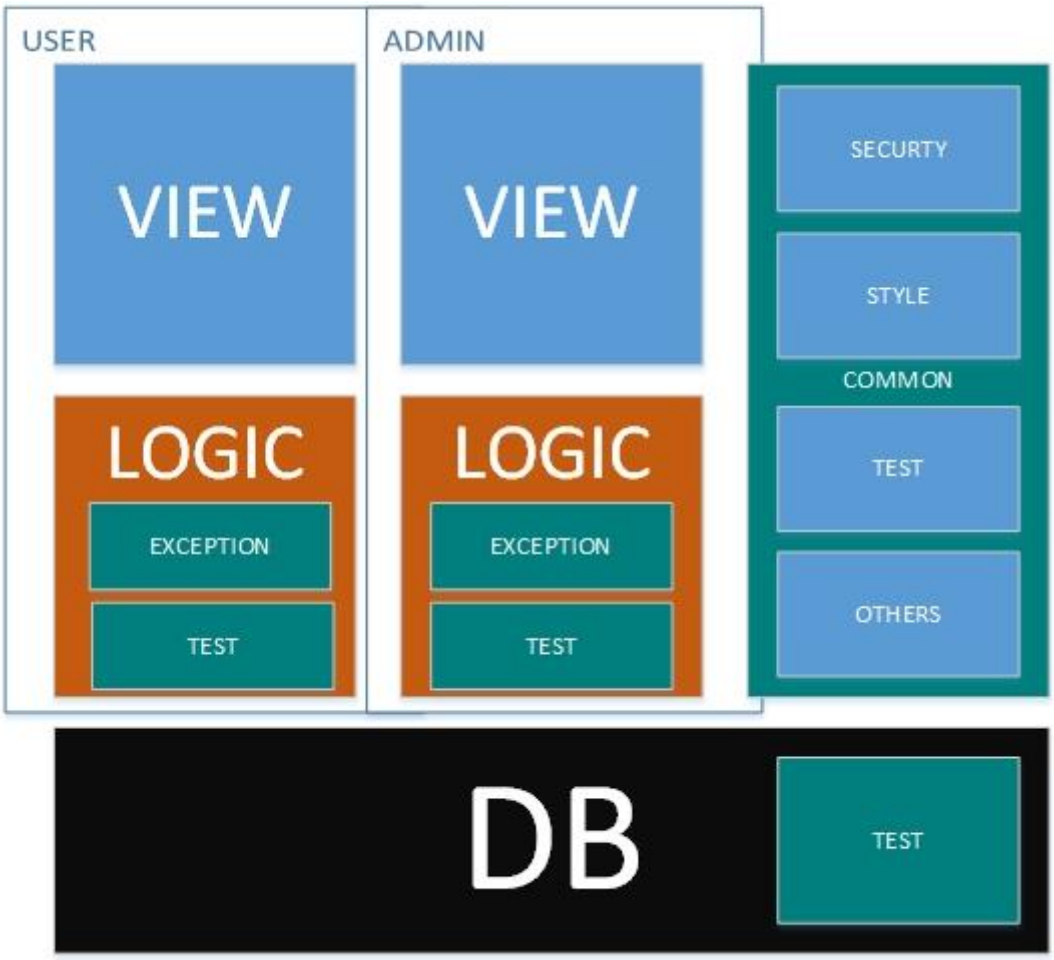

图 4：开发版本架构

3.4：命名规范

逻辑层：getXXX.php/setXXX.php/deleteXXX.php/updateXXX.php
DB 层：DBXXX.php