浅谈前后端分离

张佳瑶，3170103240

浙江大学计算机科学与技术学院软件工程专业

摘　要: Web 应用开发模式由前后端耦合发展为前后端分离。本文介绍了前后端分离开发模式的概念，分析前后端分离的优缺点，结合实际代码，展现分离式开发的具体实现。

关键词: 前端；后端；前后端分离；软件开发；Web 应用

1 引言

软件系统架构是一系列抽象模式，用于指导软件系统各方面的设计。从要解决的问题出发，确定系统边界，切分系统组件，设立沟通机制，设计实现过程，最后联系组件，合并成为一个整体，输出预期的效果。

一个网站的交互风格会大大影响系统性能。采用好的架构可以节省软件项目构建与维护的人力成本，使得每次变更都短小简单、易于实施且避免缺陷。一个好的架构需要具备模块化设计、可重组、高内聚、低耦合等特征。

随着新技术的出现，快速发展的编码标准和改进的基础架构导致了 Web 开发策略的发展。Web 应用程序体系结构通过将各个部分保持在一起来促进成功的 Web 开发。前端和后端是 Web 应用程序体系结构的两个主要子部分。

自从 Edsger W. Dijkstra 中提出关注点分离之后[1]，就一直在软件领域教授和实践应用程序分离背后的基本原理。这个概念很简单：将计算机程序分为几个部分，每个部分分别解决应用程序的不同问题。这种所谓的“关注”可以像命名实例化的类一样简单，也可以像针对平台的细节一样广泛。通常，关注点分离可以归结为在应用程序代码库的每个层，服务或模块之间创建定义明确的接口。

软件应用正在往兼容多端、高性能的方向发展。观览软件应用开发模式的发展史，基本上是前后端从紧密联系到前后端端分离的过程。在初期，前后端高度耦合，数据展现和业务逻辑混杂在一起，代码高度耦合，难以维护。后来出现了 MVC 开发模式，前后端有了简单的分工。

2 前端和后端

软件开发人员习惯于使用“前端”和“后端”来分别描述客户端和服务器端在应用程序上的工作。前端是用户可以看到的部分，而后端是支持它的基础结构。

2.1 前端

网站的前端是用户在浏览器的帮助下可以看到并进行交互的部分。前端开发也被称为“客户端编程”。前端涵盖了用户体验中直接涉及的所有内容。例如，文本，颜色，图像，导航，菜单，图标等。前端开发中使用的基本语言有 HTML，CSS 和 JavaScript。除此之外，前端开发的框架有 Bootstrap、Angular 等；前端开发的库有 JavaScript 库，如 React、Vue、jQuery。

2.2 后端

后端是网站中远离用户视线的部分。后端开发也称为“服务器端编程”，以有组织的方式进行数据管理和交互。后端开发人员需要监督软件产品或网站的功能和性能，以确保一切正常。创建后端需要 Java、Hibernate、SQL 数据库（MySQL、PostgreSQL 等）或 NoSQL 数据库（Mongo、Neo4j、Couchbase 等）和 Spring MVC、JSF 框架。它是网站的一部分，是不与用户直接联系的软件部分，无法被查看和交互。用户可以通过前端应用程序间接访问后端设计人员开发的零件和特性。后端还包括诸如编写 API，创建数据库以及在没有用户界面甚至是科学编程系统的情况下使用系统组件等活动。

前端和后端的通信方式有 Ajax，WebSocket，CORS。

2.3 前后端分离

前端和后端这两个组件对于开发完整的应用程序都是必需的。前后端分离是整个行业的规范，它造成了团队动态。这种概念上的分歧已经演变成为每个角色创建专门的开发人员角色。角色之间严格区分，可以帮助我们避免做不想做的工作。

前端开发和后端开发需要不同的技能。前端侧重于设计和可用性，后端侧重于逻辑与问题解决。如今，随着越来越复杂的客户端应用程序的出现，为了给用户提供响应快速、沉浸式、支持脱机的体验，大量功能从后端转移到了前端。前端与后端的重要性之间的差距正在缩小。前端和后端开发人员都需要考虑很多方面的问题，例如可测试性、可维护性、持久性、异步性、状态管理和 API 设计。

3 前后端分离分析

3.1 前后端分离优点

3.1.1 API 整合复用

如今，软件应用需要适应大量设备的可用性，如网站、Android 应用程序、iOS 应用程序。前后端分离技术使得网站基于 API 接口实现。API 接口可以重用，满足多终端设计与运行需求，大大简化了开发人员的代码管理工作。对于同一个应用，一套后端代码，可以支持多个版本的前端应用，极大降低了开发成本。功能强大且高性能的 Web 浏览器具有增强的处理功能，有助于在 Web 应用程序开发模型中将前端和后端分离之后的无缝衔接。

3.1.2 模块化

前后端分离的开发模式有助于模块化设计。前后端开分离开发模式中的组件和模块都是独立的，模块之间的依赖关系更加少。

前后端模块具体实现形式的更改不会影响对方，即 Web 应用程序的后端模块中的更改不会影响前端部分，反之亦然。每个模块都可以扩展，独立升级，因此，更换或者更改任何模块或组件都是十分便捷，可以以不同的速度增加前端和后端的资源。

通过将每个模块的实现隐藏在接口后面，开发人员可以在代码的不同部分进行工作，而无需了解其他区域的内部。他们不需要对其他部分进行任何更改，只需要在共享的接口界面中进行修改即可。每个应用程序层可以以不同的方式暴露给其他层意味着，开发人员可以自由地零散地更新事物，而不必进行大而昂贵的休整或丧失功能。

最终，开发团队中的成员可以独立开发，不会覆盖或破坏其他人的开发工作。

3.1.3 快速开发和部署

前端和后端分离可以更好地映射到敏捷软件开发的软件开发方法上。一个 Web 应用程序可以被分为多个相互联系、独立运行的小项目，分别去完成。各个团队在项目上并行工作，助于快速同步开发 Web 应用程序，从而快速实现应用程序部署。在这个过程中，软件一直处于可使用状态。开发人员的工作不会相互依赖。因此，无论优化工作何时完成，应用程序都可以在旧的 API 版本上进行测试和部署，而不必等待其他人的完成。

3.1.4 技术专深

软件系统通常采用多层层次结构，每一层都使用复杂的技术。因此，在多层开发环境体系结构中，编程开发已经演变成一个需要团队协作的过程。

为了创建一个复杂的系统，需要每种技术类型的专家。前后端解耦可以把前后端的开发

交给不同的项目组，有助于获得相应技术专家的程序员，使开发更加专业。对于预算充裕的大型项目，可以与各个领域的专家组成一个团队，包括设计师，Web 开发人员，移动开发人员，后端开发人员，数据库开发人员等。每个开发人员都是他们一个研究方向上的专家，可以在他们自己的一方执行任何具有挑战性的任务。

同样，前后端解耦可以消除对彼此可能施加的技术选择的限制。因此，在这样的开发环境中，前后端分离可以使开发过程顺利进行。

3.2 前后端分离缺点

松散耦合的前后端结构有许多优点，但是也存在一系列缺点。

3.2.1 沟通鸿沟

前端和后端的划分会造成沟通鸿沟。

使用 API 进行通信和代码管理会增加团队文档的管理负担。

同时，团队之间沟通效率低下。前端和后端两个团队都无法获得有关对方变更的信息。前端开发人员并不总是完全了解后端和 API 的变化，反之亦然。沟通不畅或对目标，范围，条款等的误解可能会引起问题，在最坏的情况下可能会导致开发失败。

在前端和后端的集成时，沟通鸿沟可能会导致错误，最终团队开发进度延迟。

3.2.2 额外开销

当考虑到大型项目时，在客户端和服务器端都需要解决一些任务。在这种情况下，开发人员需要从应用程序的一个模块跳到另一个模块，和不同模块的人交流沟通，从而导致额外的开销。

3.2.3 失去安全优势

前端和后端分离会失去许多安全优势。例如，前后端分离会需要公开 API，公开 API 会受到很多形式的攻击。

4 前后端分离具体实现

Web 应用的开发模式经过几代演变，从纯 Servlet 模式到纯 Jsp 模式，再发展到 Model 模式，最后发展成为前后端分离模式。[4]

在前后端未分离时，软件开发采用 Jsp + Servlet 的开发模式。JSP 负责视图层渲染及交互；Servlet 负责接收表单参数，按请求方式手动封装请求参数进行请求，处理业务逻辑与页面导航。一个大体的流程是，编写 JSP 页面中的 HTML、CSS、JavaScript，重写 Servlet 的

service()方法，配置 web.xml。

前后端分离后，一个 Web 应用分为前端项目和后端项目。前后端分离的核心思想是前端通过 ajax 调用后端的 Restful API，使用 JSON 数据进行交互。前端可以是 App，小程序，或 H5 等。后端项目以 Web 服务的形式呈现，运行在云服务器上，访问数据库，以 JSON 的方式提供 API 接口。

前端模型构建视图层、接口层，其中视图层包括组件、路由模块，接口层包括所有具体的业务接口。服务端使用 nginx[2]作为静态资源代理。采用 Node.js 作为前端和后端服务的中间层。并且服务端可以根据应用的性能要求，采用负载均衡的方案以达到高性能的要求，从而快速响应客户端的请求。[3]

用户通过界面输入请求。然后将其验证并传达给服务器，服务器从数据库中提取必要的数据并将其发送回用户。

以“用户登陆”模块为例，以代码展现前后端分离的具体实现。前端通过 vue 实现模块化和组件化，通过 vue-router 管理用户请求和页面跳转；后端通过 RESTful 架构实现前后端分离架构；在前端和后端之间引入 Node.js 作为中间层。前端与后端定义 JSON 数据结构，一般包含两部分：元数据与返回值。其中，元数据表示操作是否返回与返回消息等，返回值对应服务端方法所返回的数据。JSON 响应结构可以如下：

```
{
    "meta": { "success": true,  "message": "ok" },
    "data": ...
}
```

4.1 前端登陆界面

login.vue 为前端“用户登陆”模块代码：

```
<template>
  <div>
    <el-form :model="ruleForm" ref="ruleForm">
      <el-form-item label="账号" prop="username">
        <el-input v-model="ruleForm.username"/>
      </el-form-item>
      <el-form-item label="密码" prop="password">
```

```
      <el-input type="password" v-
model="ruleForm.password"/>     </el-form-item>
     <el-form-item>
      <el-button @click="login">登录</el-button>
     </el-form-item>
   </el-form>
  </div>
</template>

<script>
export default {
  data() {
    return {
      ruleForm: {
        username: "",
        password: ""
      }
    };
  },
  methods: {
    login() {
      this.$axios
        .post("/user/login", this.ruleForm)
        .then(ret => {
          this.$showMessage(ret.data);
        })
        .catch(err => {
          console.log(err);
        });
    }
  }
};
</script>
```

前端通过 axios 进行 ajax 请求，将登陆信息（账号、密码）放在请求体中，传输给给后

端登陆模块接口。前端控制页面，根据元数据与返回值，确定页面呈现内容。在“用户登陆”模块中，前端请求后可以呈现出登陆成功和登陆失败两种场景。

4.2 后端登陆接口

Spring MVC 是前端和后端接口实现的主要框架，对前后端分离模式开发研究具有重要意义。Spring 框架体现了分层设计的思想。Spring 中的面向切面编程（aspect-oriented programming, AOP）可以将遍布应用各处的功能分离出来形成可充用的组件。系统有许多不同的组件组成，每一个组件负责一块特定功能。AOP 可以将服务模块块化，并以声明的方式将他们应用到他们需要影响的组件中去。系统服务器横切关注点，关注日志、事务管理、安全等方面。

一般将领域模型划分为终端显示层、Web 层、Service 层、Dao 层。

终端显示层：各个端的模版渲染并执行显示的层。显示层对象采用 VO 后缀。

Web 层：对访问控制进行转发，各类基本参数校验，或者不复用的业务简单处理等。

Service 层：实现具体的业务逻辑。业务层对象采用 BO 后缀。

Dao 层：数据访问层，与底层 MySQL、Redis 进行数据交互。数据访问层对象采用 DO 后缀，采用 Query 后缀作为查询请求，采用 DTO 后缀作为数据传输对象。

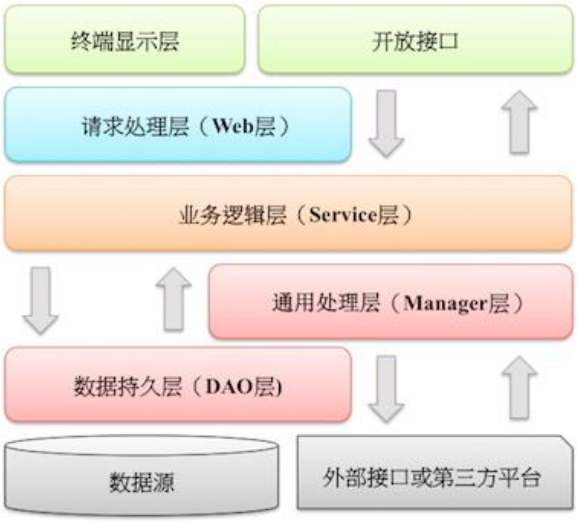

图 4-2-1 后端层次划分

后端项目采用 SSM 框架实现。后端登接口的代码如下：

```
@PostMapping("/user/login")
public ResultMessage login(@RequestBody User user,
HttpServletRequest httpServletRequest) throws
```

```
JsonProcessingException {
    ResultMessage resultMessage;
    User user1 = userService.getUser(user);
    if (user1 != null) {
        resultMessage = new
ResultMessage(StatusCode.SERVICE_RUN_SUCCESS);
    } else {
        resultMessage = new
ResultMessage(StatusCode.USER_LOGIN_ERROR);
    }
    return resultMessage;
}
```

登陆接口 login 接收到请求体并且封装成 User 对象，进行登陆的业务处理。查询底层数据库，验证账号、密码。如果验证成功，返回登陆成功的 JSON 数据；如果验证失败，返回登陆失败的 JSON 数据。

5 小结

前后端分离有一些优点和缺点。考虑到当前的开发方案，前后端分离模式的开发是一个种首选方式。前后端分离不仅可以避免后端的重复开发，又可以避免前后端功能的相互依赖，从而可以实现前后端工程师并行进行开发，降低了程序的开发周期，提高程序开发效率。

参考文献

[1] Dijkstra E W. On the role of scientific thought[M]//Selected writings on computing: a personal perspective. Springer, New York, NY, 1982: 60-66.
[2] 肖明魁. 基于 Nginx 负载均衡技术初探[J]. 科技展望, 2015, 36: 125.
[3] 杜艳美, 黄晓芳. 面向企业级 web 应用的前后端分离开发模式及实践[J]. 西南科技大学学报 (自然科学版), 2018, 33(2): 83-87.
[4] 孟祥双. 前后端分离式 WEB 应用开发研究[J]. 电子元器件与信息技术, 2019, 3(6): 40-43.