如何用 Swagger 消除前后端分离的障碍

前言

随着后端技术的日渐成熟和前端框架的异军突起，前后端分离几乎已经企业开发必须要选择的方向。但是采用了JSP、Struts2、SpringMVC等技术的项目在实现前后端分离时，由于Web容器的使用，使得前端开发人员无法专注于前台展示，分离起来困难重重。RESTful架构的引入，让电脑端、移动端、第三方系统的数据交互和流程控制更加便捷，也使得前后端分离成为可能。但是随着项目的推进，众多的API如果管理不当，在项目集成时，又将陷入泥潭。而Swagger的出现专注于为API提供可视化文档，便于前后端开发人员对接。

前后端分离很重要？

对于项目

前端界面注重展示效果，希望有更好的交互性和快速的响应。后台服务专注于提供数据，希望有稳定的性能，需要确保数据的安全，降低系统的瓶颈。

在业务逻辑复杂的系统里，最难维护的就是那些前后端混杂在一起的代码。我所在的项目组，搜索业务的代码原来是用JSP写的，代码风格相对随意，在数次艰难的bug修复后，终于全部废弃了。

对于公司

前后端界定不清，公司的招聘可能难度更大。full stack的工程师可遇而不可求，就算直接培养也很难使其在各个领域都成为大神。而且full stack意味着他掌握的业务线越长，一旦离职，会对整个团队造成较大的影响。职责清楚，分工明确，意味着公司招聘更简单。

对于个人

前端人员有着自己的开发流程，有更适合的构建工具和测试方法，他们不希望用Maven构建项目、不希望用Eclipse编写代码，往往需要把用户体验放在第一位。

后端人员也有自己的开发习惯，他们不想用Gulp构建项目，也不会选择WebStrom调试代码，整天围绕着数据存贮与提取，数据计算，数据安全，往往把数据放在第一位。

> 前后端分离就是为了提高效率，提高项目的开发效率，提高公司的招聘效率，提高开发人员的学习效率。然而，以API形式解耦前后端开发过程，通过RESTFul方式通信，项目集成将是绕不开的话题。开源项目swagger正是在这样的背景下因运而生。

Swagger 初步认知

Swagger项目

Swagger是很多产品的总称。包含最核心的规范Swagger Specification，编辑器Swagger Editor，图形界面Swagger UI，代码生成器Swagger Codegen，成熟的产品SwaggerHub等。Swagger的主要作用是描述RESTful API，生成交互式文档，便于前后端开发人员查看请求信息和响应数据。

Swagger与OpenAPI

Swagger项目最初由非营利组织Wordnik于2010年发起的， 之后于2015年转交给了SmartBear，这家SmartBear公司专注于开发测试和性能工具。随后SmartBear又在Linux基金会下创建了一个OpenAPI Initiative(OAI)项目，并捐献出了Swagger。由于Google，IBM，Microsoft等公司对OpenAPI Initiative的持续贡献，OAI日渐成熟，Swagger 又被称之为OpenAPI。我们目前常用Swagger 的版本为Swagger 2.0，它的下一版不是Swagger 3.0，而是OpenAPI 3.0

Swagger 快速上手

要了解Swagger，必须知道一些常用的Swagger 规范，官方建议使用JSON格式或者YAML语法来编写这一规约，我更倾向于YAML，这种格式更容易书写和阅读。

Swagger Editor 编辑器

首先我们需要用到Swagger Editor，你可以直接使用在线版本http://editor.swagger.io/，或者在本地使用。如果电脑已经安装好了Node.js，用下面的命令即可获得本地编辑器。

```
git clone https://github.com/swagger-api/swagger-editor.git
cd swagger-editor
npm start
```

Node.js不是必须的，你也可以直接下载Swagger Editor的源代码

https://github.com/swagger-api/swagger-editor。

解压以后，用本地浏览器打开 `./dist/index.html` 即可。

> Swagger Editor 只是一款编辑器，学习Swagger不一定要用到它。你也可以用记事本或者Eclipse等工具编辑Swagger规约。不过Swagger Editor 有如下几个有点：1. 代码提示与代码高亮；2. 语法检查； 3. 实时预览，左边写code，右边实时生成UI。

Swagger Specification 规约

下面是我用Swagger规范写的一个”Hello World”，使用Swagger Editor编辑，可以很快得到下面的文档，而且可以在右侧看到生成的图形化界面。

首先是YAML格式，如swagger.yml：

```
swagger: '2.0'
info:
  version: 1.0.0
  title: My Hello world
  description: description example
host: test.com
basePath: /
schemes:
- http
paths:
  /hello:
    get:
      summary: "hello, world"
      description: "My first swagger"
      parameters:
      - in: query
        name: name
        description: The name of the person
        required: true
        type: string
      responses:
```

```
        200:
          description: OK
```

如果选择JSON格式，如 swagger.json ：

```
{
    "swagger": "2.0",
    "info": {
        "version": "1.0.0",
        "title": "My Hello world",
        "description": "description example"
    },
    "schemes": [
        "http"
    ],
    "host": "test.com",
    "basePath": "/",
    "paths": {
        "/hello": {
            "get": {
                "summary": "hello, world",
                "description": "My first swagger.",
                "parameters": [
                    {
                      "in": "query",
                      "name": "name",
                      "description": "The name of the person",
                      "type": "string"
                    }
                ],
                "responses": {
                    "200": {
                        "description": "A list of Person",
                    }
                }
            }
        }
    }
}
```


对比发现，YAML格式看起来更舒服，写起来也更加便捷。当然，已经有很多工具可以实现YAML-JSON之间的转换，用自己喜欢的风格即可。规范的内容本身很容易理解，配合Swagger Editor很容易能写出对用的API文档。如果需要更详细的说明，可以在官网的doc页面找到。

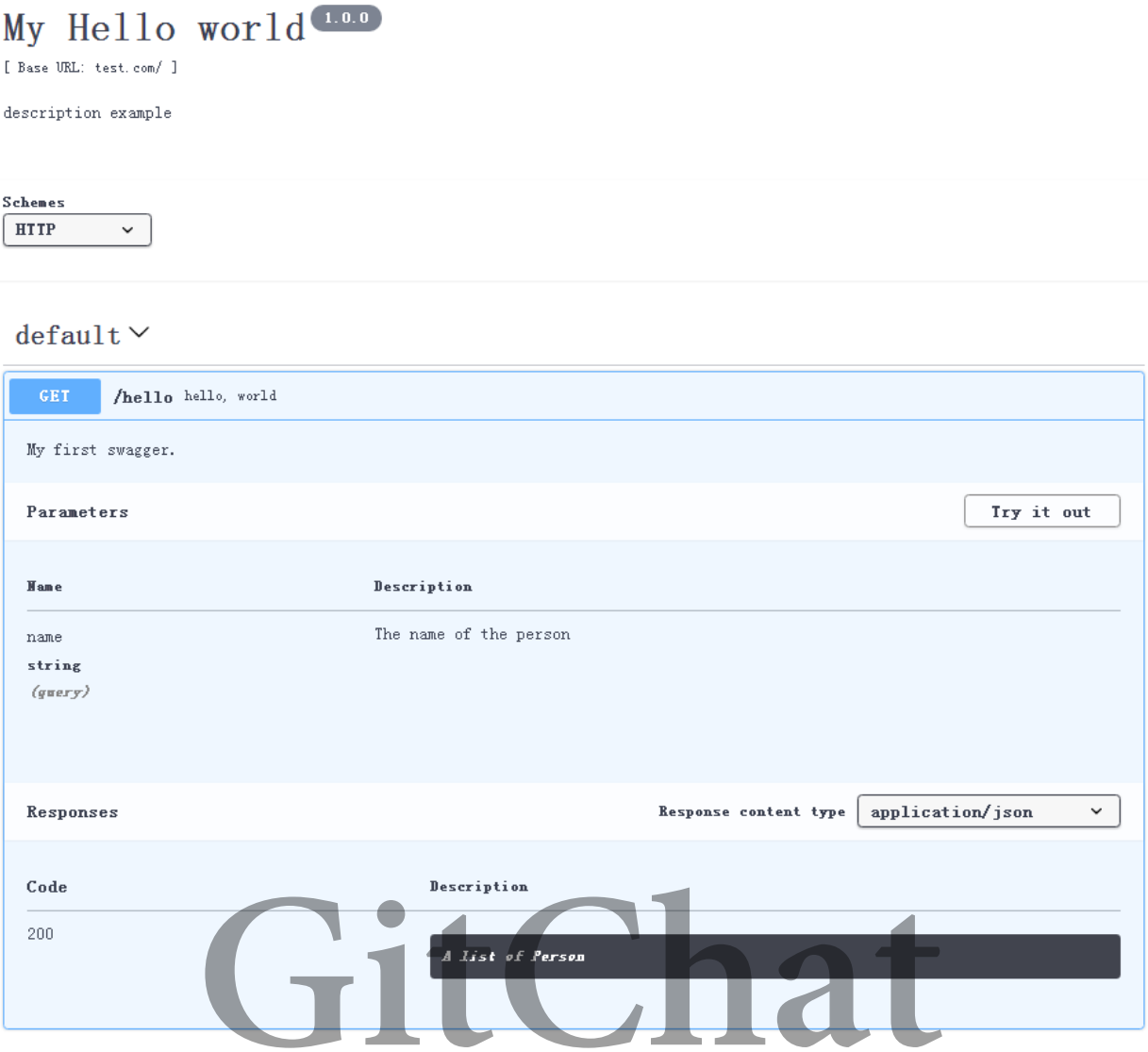
Swagger UI 交互式图形化界面

刚才得到的Swagger规约可以方便地在团队间编辑，传输，但是不够直观，于是有了Swagger-UI。它是为基于Swagger规范的API生成基于纯粹HTML,javascript,css的在线可视化文档，同时也能成为API的在线测试工具。你可以在GitHub上得到其源代码https://github.com/wordnik/swagger-ui。其中 `./dist/*` 中的文件是生成好的文件，可以直接使用，你也可以修改源代码，然后使用Node运行。可以看到最新的版本于2017-09-15发布，该版本最大的优势就是支持 `OpenAPI 3.0`

我们得到的文件 `swagger-ui/dist/index.html` 中有如下代码片段，将其中的 `url` 替换成自己的文件，如 `url: "swagger.yml"` ，用浏览器打开index.html，就能看到漂亮的文档说明。

```
<script>
window.onload = function() {

  // Build a system
  const ui = SwaggerUIBundle({
    url: "http://petstore.swagger.io/v2/swagger.json",
    dom_id: '#swagger-ui',
    deepLinking: true,
    presets: [
      SwaggerUIBundle.presets.apis,
      SwaggerUIStandalonePreset
    ],
    plugins: [
      SwaggerUIBundle.plugins.DownloadUrl
    ],
    layout: "StandaloneLayout"
  })
  window.ui = ui
}
</script>
```



Swagger与Spring Boot

如果你的项目运用了Microservice，引入了Spring Boot，构建的RESTful API通过手工管理简直是噩梦。如今，Swagger与Spring Boot的无缝对接，简直就是不愿写文档的猿类的福音。

需要的依赖

在pom.xml中需要引入支持Swagger所需要的依赖包。

```
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger2</artifactId>
    <version>2.2.2</version>
</dependency>
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger-ui</artifactId>
```

```
        <version>2.2.2</version>
    </dependency>
```

需要的配置

将Swagger整合到项目中，需要在启动类 `Application.java` 同级目录下创建一个配置类，名称任意，如 `Swagger.java`

```
    @Configuration
    @EnableSwagger2
    public class Swagger {
        @Bean
        public Docket createRestApi() {
            return new Docket(DocumentationType.SWAGGER_2)
                    .select()

    .apis(RequestHandlerSelectors.basePackage("com.gitchat.controller
    "))
                    .paths(PathSelectors.any())
                    .build();
        }

    }
```


这里需要制定一个basePackage()，不然整个项目中所有的API都会生成文档，包括Spring自己的service。

测试一下

启动Spring Boot程序后，访问：

http://localhost:8080/swagger-ui.html

将看到类似Swagger UI的效果。点击 `Try it out` 可以测试API的是否正常工作。

Swagger 与你有多近

基于前后端分离的契约，无论你是后端开发，前端开发，测试人员，你都可以体验到Swagger的众多好处。

- 生成交互式文档：一般开发人员容易忽视API文档，程序员最不喜欢的就是写文档，但是总是抱怨这个维护的程序怎么没有文档。

- 人类可读与机器可读：选择JSON和YAML格式作为Swagger的规范格式不是偶然的结果，这样可以让方便更多的用户编辑文档，分享分档，可以很方便地在各种编程语言之间直接调用或相互转换。

无论是手动编写Swagger规范，还是代码自动生成Swagger规约，漂亮的文档只是帮助我们更好的完成开发任务，Swagger不是必须的，但，可能是必要的。

前后端分离了，接下来，就看你的了！