WireMock 简单入门

一、背景

最近公司的项目需要重构，采用前后端分离模式。由于前后端开发人员开发效率和进度的不同，前端人员在编写调用接口代码时需要伪造数据。这意味着 N 个前端人员需要维护 N 份数据，且每个前端人员可能对业务理解不同，伪造出的数据存在偏差，因此在真正与后端联调接口时会出现问题。

为了解决这一问题，我们可以使用 WireMock。

二、介绍

2.1 简单介绍

WireMock 是基于 HTTP 的模拟器。它具备 HTTP 响应存根、请求验证、代理/拦截、记录和回放功能。

当开发人员的开发进度不一致时，可以依赖 WireMock 构建的接口，模拟不同请求与响应，从而避某一模块的开发进度。

2.2 下载文件

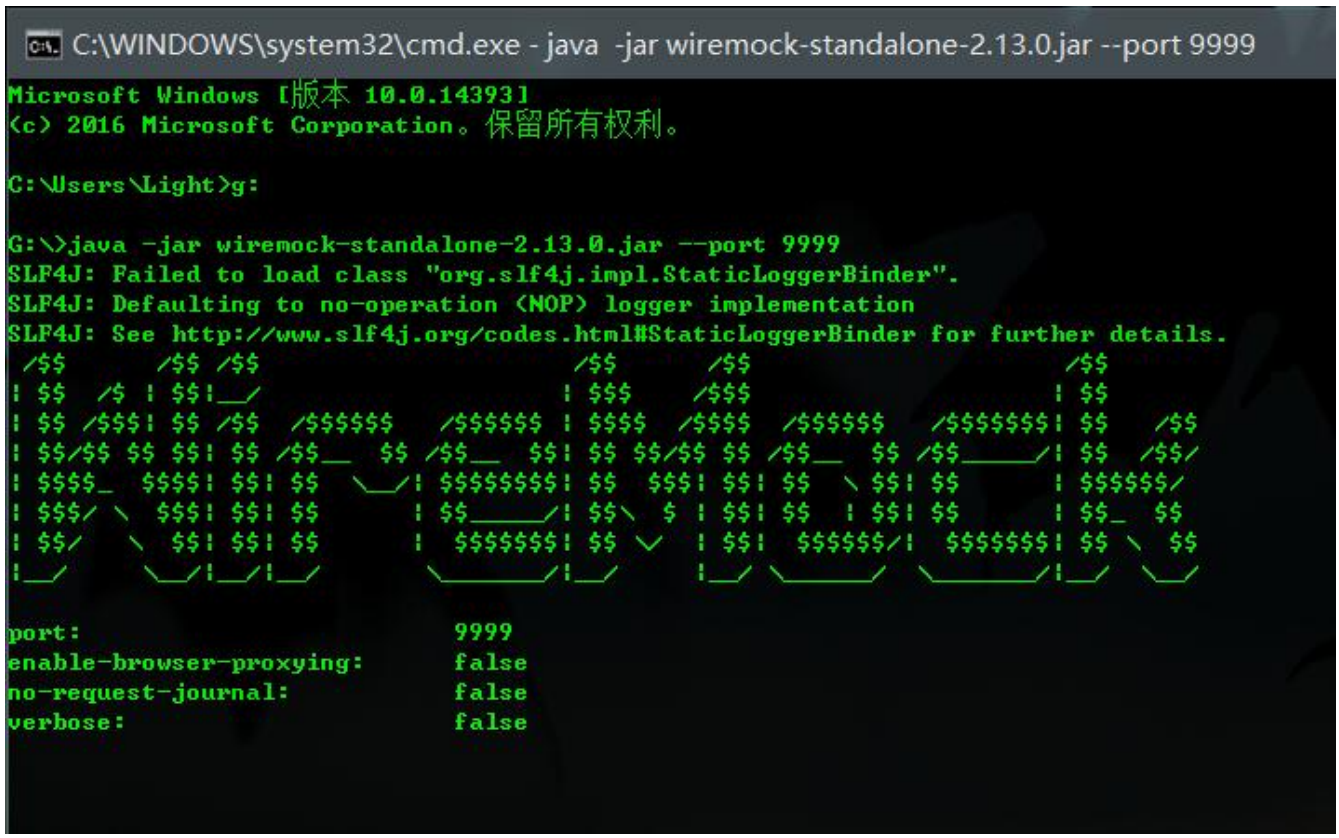
点击 http://wiremock.org/docs/running-standalone/ 下载启动 WireMock 服务的 jar 包。

2.3 启动服务

在 jar 包所在目录执行如下命令：

```
java -jar wiremock-standalone-2.13.0.jar --port 9999
```

启动后，如下图：



更多参数请参考官方文档。

三、编写响应

WireMock 服务启动后，它现在还只是一个空壳，我们需要向服务添加请求规则与请求响应。

创建一个 maven 项目，编写 WireMock 客户端代码。

3.1 添加依赖

```
<dependency>
    <groupId>com.github.tomakehurst</groupId>
    <artifactId>wiremock</artifactId>
    <version>2.13.0</version>
</dependency>
```

3.2 编写响应

```
import static com.github.tomakehurst.wiremock.client.WireMock.*;

public class App {

    public static void main(String[] args) {
        // 连接 9999 端口
        configureFor(9999);
        // 删除旧的规则
        removeAllMappings();

        stubFor(get(urlPathEqualTo("/user/1"))
                .willReturn(aResponse()
                .withStatus(200)
                .withBody("{\"id\":1,\"name\":\"Jack\"}")));

    }
}
```

补充：

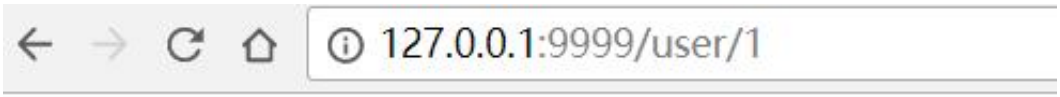
get()：get 请求

withStatus：返回状态

withBody：返回数据

方法执行后，我们模拟的请求与响应就添加到 WireMock 服务中了。

打开浏览器访问 http://127.0.0.1:9999/user/1，效果图如下：




```
{
    "id": 1,
    "name": "Jack"
}
```


四、优化

在实际业务代码中，调用某个接口不可能只返回简单的 JSON 字符串。如果将这些字符串写到上文的代码中，当需要修改数据结构时将是一个灾难，因此我们可将返回的数据写入到文件进行维护。

4.1 新建文件

在项目的 src/main/resources 目录下创建名为 user.txt（自定义），内容如下：

```
{
    "id":1,
    "name":"jack",
    "age":18,
    "birthday":"2018-01-01"
}
```

4.2 添加依赖

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
    <version>4.3.12.RELEASE</version>
</dependency>
<dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>2.6</version>
</dependency>
```

在测试的代码中需要用到上述依赖的 API，如果读者已自定义好读取文件的 API，可以略过该步骤。

4.3 修改代码

```
import static com.github.tomakehurst.wiremock.client.WireMock.*;
import java.io.IOException;
import org.apache.commons.io.FileUtils;
import org.springframework.core.io.ClassPathResource;


public class App {

    public static void main(String[] args) throws IOException {
        // 连接 9999 端口
        configureFor(9999);
        // 删除旧的规则
        removeAllMappings();
        // 添加请求规则和请求响应
        mock("user.txt","/user/1");

    }

    private static void mock(String fileName, String url) throws
IOException{
        // 加载文件
        ClassPathResource resource = new ClassPathResource(fileName);
        // 读取内容
        String data = FileUtils.readFileToString(resource.getFile(), "UTF-
8");

        stubFor(get(urlPathEqualTo(url))
                .willReturn(aResponse()
                    .withStatus(200)
                    .withBody(data)));
    }
}
```

执行方法，打开浏览器访问 http://127.0.0.1:9999/user/1，效果图如下：


```
{
    "id": 1,
    "name": "jack",
    "age": 18,
    "birthday": "2018-01-01"
}
```


当我们需要为前端模拟新接口时，只需对 mock 方法传入不同参数运行即可。

注意：上文的 **mock** 方法只是针对 **get** 请求进行模拟，如需模拟不同方法的请求，请参考下边的资料。

五、参考资料

- http://wiremock.org/docs/java-usage/ 官方文档