SpringCloud之九

服务链路追踪Spring Cloud Sleuth

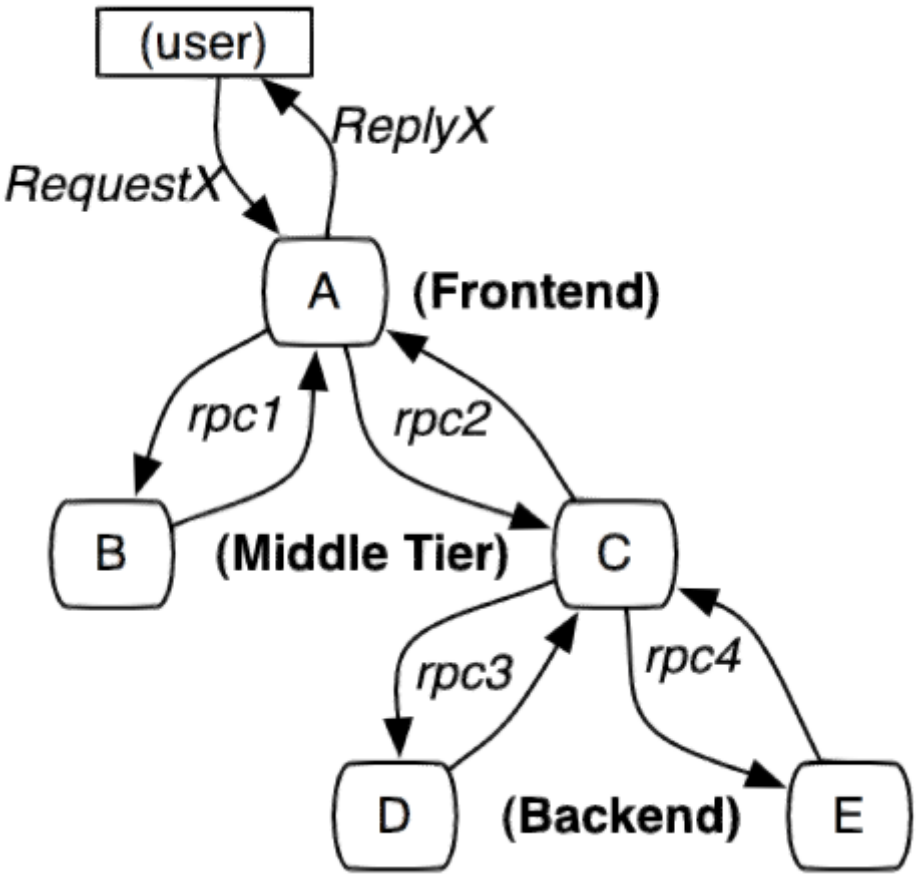
一、Sleuth介绍

Spring Cloud Sleuth是Spring Cloud的分布式跟踪工具。它借鉴了Dapper，Zipkin和HTrace。

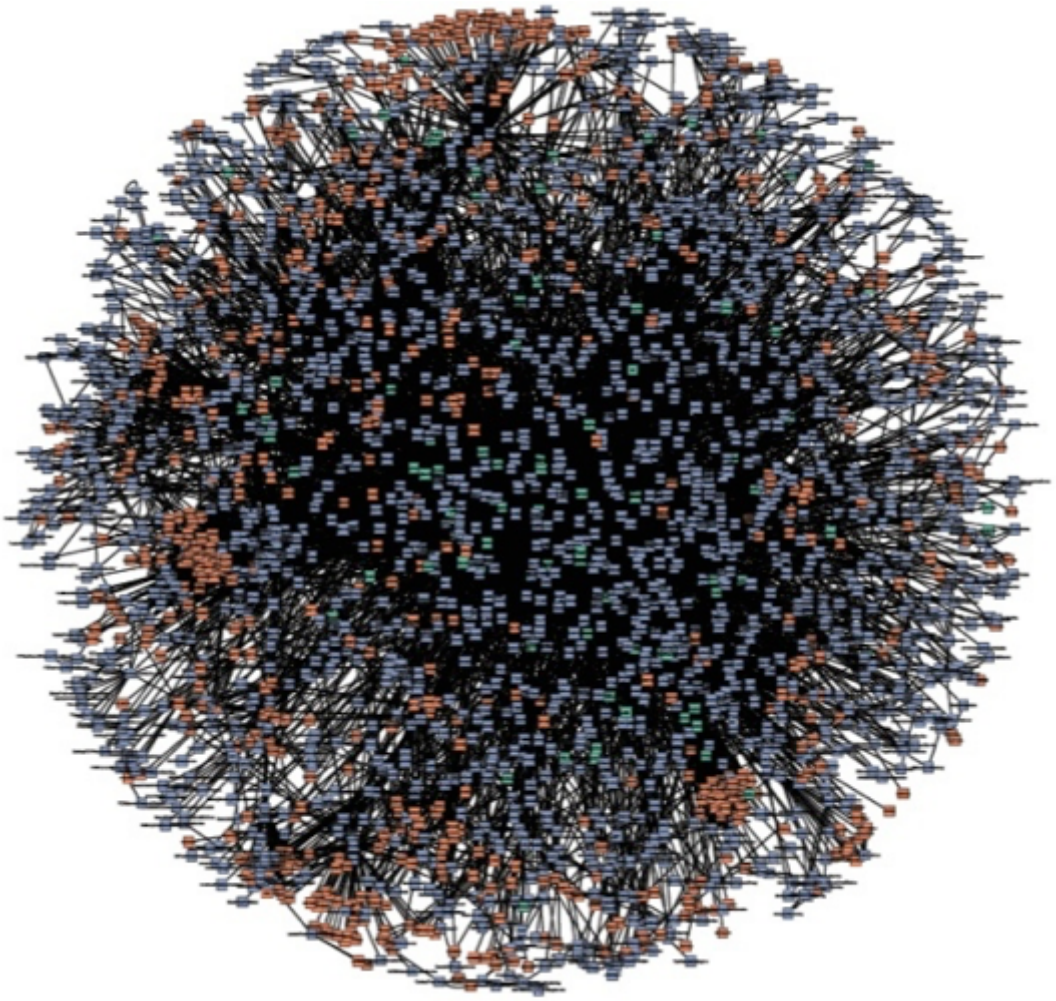
在官网中：

服务追踪分析

微服务架构上通过业务来划分服务的，通过REST调用，对外暴露的一个接口，可能需要很多个服务协同才能完成这个接口功能，如果链路上任何一个服务出现问题或者网络超时，都会形成导致接口调用失败。随着业务的不断扩张，服务之间互相调用会越来越复杂。



随着服务的越来越多，对调用链的分析会越来越复杂。它们之间的调用关系也许如下：



术语

1、Span：基本工作单元，例如，在一个新建的span中发送一个RPC等同于发送一个回应请求给RPC，span通过一个64位ID唯一标识，trace以另一个64位ID表示，span还有其他数据信息，比如摘要、时间戳事件、关键值注释(tags)、span的ID、以及进度ID(通常是IP地址)。

span在不断的启动和停止，同时记录了时间信息，当你创建了一个span，你必须在未来的某个时刻停止它。

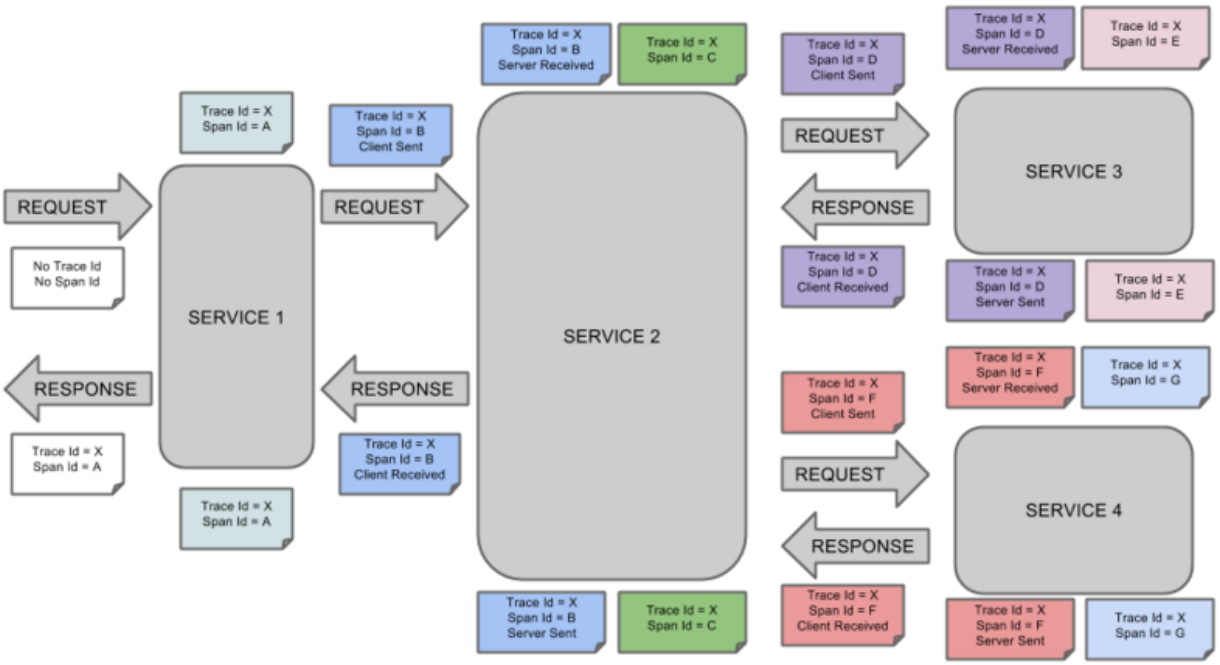
2、Trace：一系列spans组成的一个树状结构，例如，如果你正在跑一个分布式大数据工程，你可能需要创建一个trace。

3、Annotation：用来及时记录一个事件的存在，一些核心annotations用来定义一个请求的开始和结束

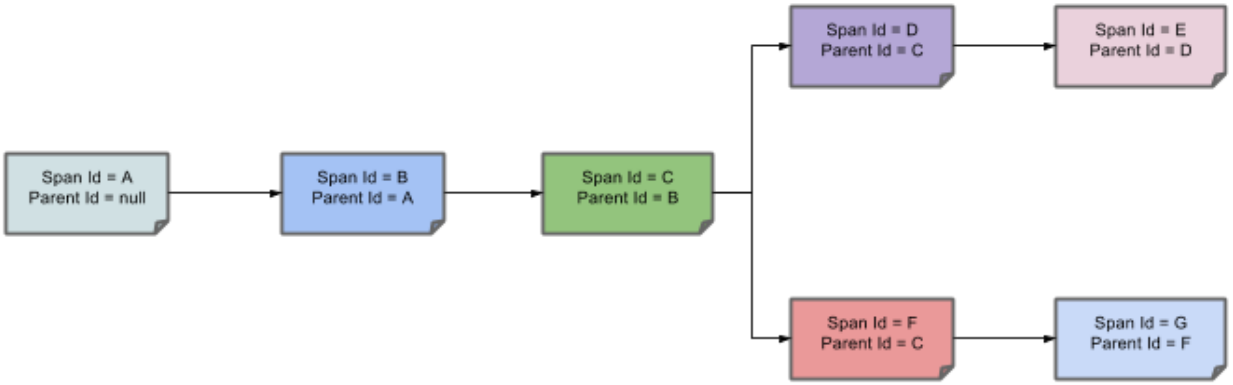
- cs - Client Sent -客户端发起一个请求，这个annotion描述了这个span的开始
- sr - Server Received -服务端获得请求并准备开始处理它，如果将其sr减去cs时间戳便可得到网络延迟

- ss - Server Sent -注解表明请求处理的完成(当请求返回客户端)，如果ss减去sr时间戳便可得到服务端需要的处理请求时间
- cr - Client Received -表明span的结束，客户端成功接收到服务端的回复，如果cr减去cs时间戳便可得到客户端从服务端获取回复的所有所需时间

将Span和Trace在一个系统中使用Zipkin注解的过程图形化：



span之间的关系



Zipkin官网：https://zipkin.io/

二、Sleuth 的使用

主要有三个工程组成:

1、zipkin-server，它的主要作用使用ZipkinServer 的功能，收集调用数据，并展示；

2、login-service，对外暴露login接口；

3、user-service，对外暴露login接口；这两个service可以相互调用；并且只有调用了，zipkin-server才会收集数据的，这就是为什么叫服务追踪了。

2.1、zipkin-server

在spring Cloud为F版本的时候，已经不需要自己构建Zipkin Server了，只需要下载jar即可，下载地址：

https://dl.bintray.com/openzipkin/maven/io/zipkin/java/zipkin-server/

当前我使用的是：

https://dl.bintray.com/openzipkin/maven/io/zipkin/java/zipkin-server/2.9.4/

也可以在这里下载：

链接: https://pan.baidu.com/s/1twL44Qyc0DrqdGeborTOVA 提取码: g6tq

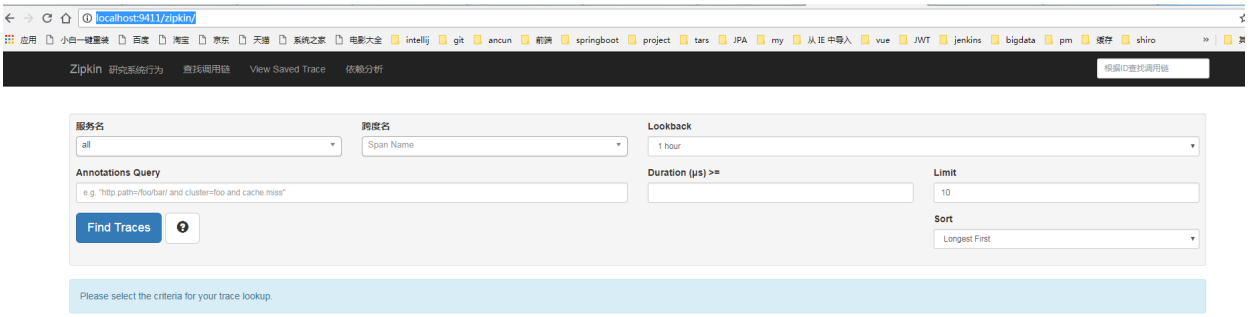
下载完成jar 包之后，需要运行jar，如下：

```
java -jar zipkin-server-2.9.4-exec.jar
```

访问浏览器 http://localhost:9411/zipkin/

发现现在最新版本应该是2.11.9版本，有需要的可以直接找这个版本，不需要拉到最后。

访问结果如图：



2.2、login-service

首先更新pom.xml文件，添加zipkin依赖，如下：

```
<dependency>
 <groupId>org.springframework.cloud</groupId>
 <artifactId>spring-cloud-starter-zipkin</artifactId>
</dependency>
```

然后再把配置文件修改，

application.yml添加spring.zipkin.base-url内容如下：

```
server:
# port: 8762
 port: 8763
# 多个服务不同的端口

spring:
 application:
 name: login-service
 zipkin:
# 通过配置“spring.zipkin.base-url”指定zipkin server的地址
 base-url: http://localhost:9411

eureka:
 client:
 serviceUrl:
 defaultZone: http://localhost:8761/eureka/

```

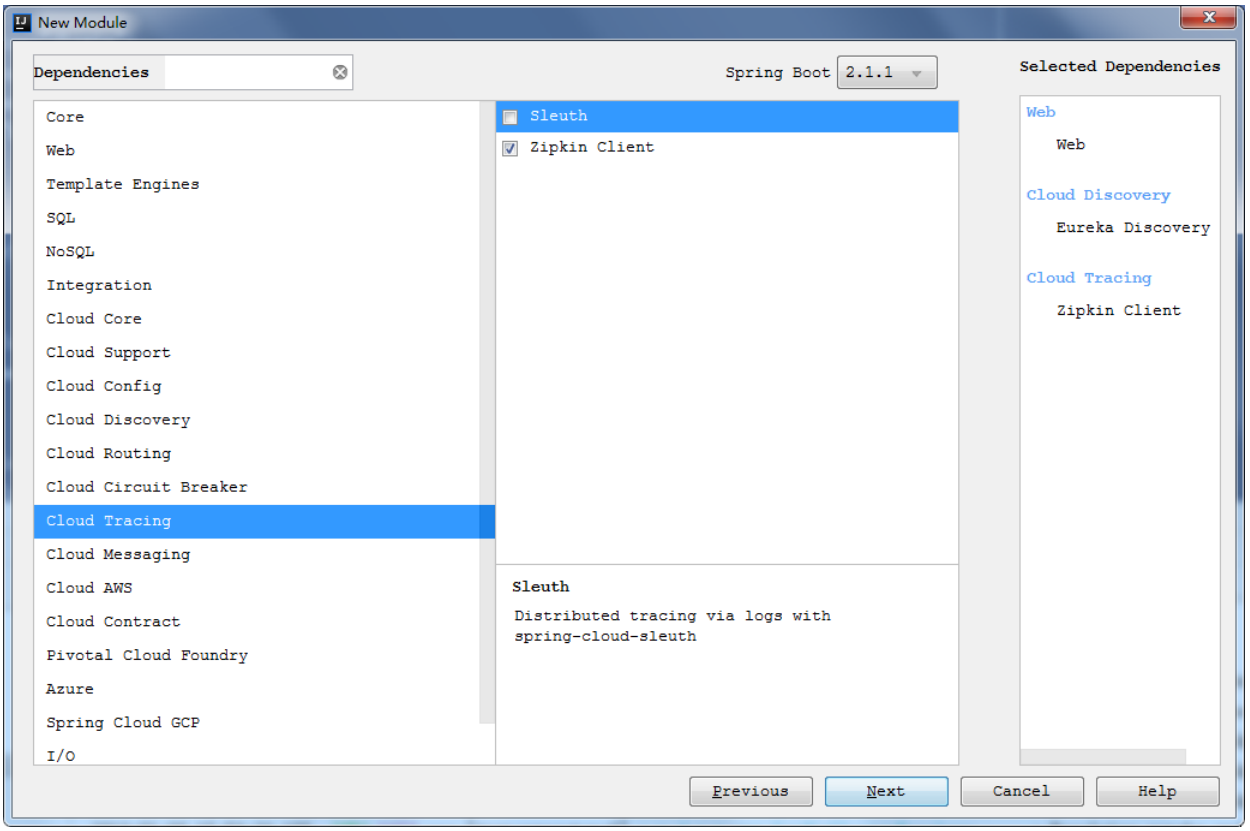
在LoginController.java类添加内容如下：

```
1  package xyz.jiangnanke.loginservice.controller;
2
3  import brave.sampler.Sampler;
4  import org.springframework.beans.factory.annotation.Autowired;
5  import org.springframework.beans.factory.annotation.Value;
6  import org.springframework.context.annotation.Bean;
7  import org.springframework.web.bind.annotation.RequestMapping;
8  import org.springframework.web.bind.annotation.RequestParam;
9  import org.springframework.web.bind.annotation.RestController;
10  import org.springframework.web.client.RestTemplate;
11
12  import java.util.logging.Level;
13  import java.util.logging.Logger;
14
15  /**
16   * @Auther: zhengfeng
17   * @Date: 2018\12\19 0019 15:48
18   * @Description:
19   */
20 @RestController
21 public class LoginController {
22   private static final Logger LOG = Logger.getLogger(LoginController.clas
s.getName());
23
24   @Value("${server.port}")
25   String port;
26
27   @RequestMapping("/login")
28   public String home(@RequestParam(value = "name", defaultValue = "jiangn
anke") String name) {
29   System.out.println(" this is login for name :" + name);
30   return "hi " + name + " ,login is success! the port is:" + port;
31   }
32
33   @Autowired
34   private RestTemplate restTemplate;
35
36   @Bean
37   public RestTemplate getRestTemplate() {
38   return new RestTemplate();
```

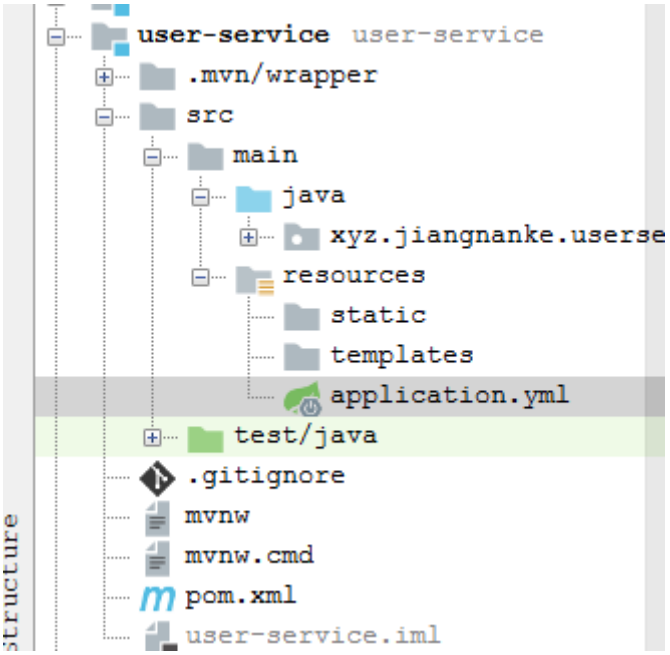
```
39   }
40   @Bean
41   public Sampler defaultSampler() {
42   return Sampler.ALWAYS_SAMPLE;
43   }
44
45   @RequestMapping("/sayHello")
46   public String callHome() {
47   LOG.log(Level.INFO, "calling trace login-service say hello ");
48   return restTemplate.getForObject("http://localhost:10180/getUser", Stri
ng.class);
49   }
50
51   @RequestMapping("/info")
52   public String info() {
53   LOG.log(Level.INFO, "calling trace service-hi ");
54   return "i'm login-service";
55   }
56
57 }
```

2.3、user-service

如创建login-service一样创建user-service，有忘记的可以查看以前的说明文档。



创建之后的得到的项目结构如下：



配置application.yml文件内容如下：

```yaml
server:
  port: 10180

spring:
  application:
  name: user-service
```

```
  zipkin:
  # 通过配置“spring.zipkin.base-url”指定zipkin server的地址
  base-url: http://localhost:9411

 eureka:
  client:
  serviceUrl:
  defaultZone: http://localhost:8761/eureka/

```

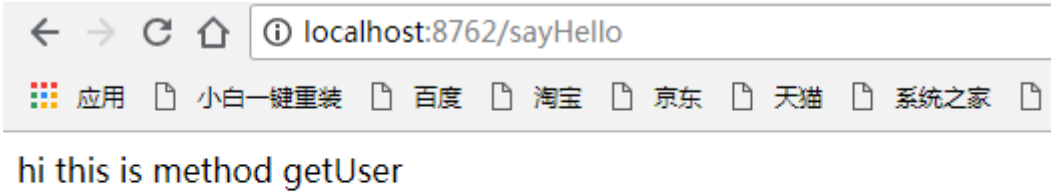
添加对外的接口UserController.java，代码如下：

```
package xyz.jiangnanke.userservice.controller;

import brave.sampler.Sampler;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.client.RestTemplate;

 import java.util.logging.Level;
 import java.util.logging.Logger;

 /**
  * @Auther: zhengfeng
  * @Date: 2019\1\8 0008 18:28
  * @Description:
  */
 @RestController
 public class UserController {

  private static final Logger LOG =
Logger.getLogger(UserController.class.getName());

  @RequestMapping("/getUser")
  public String getUser(){
  LOG.log(Level.INFO, "hi is being called UserController getUser");
  return "hi this is method getUser";
  }

  @RequestMapping("/showLogin")
```

```
30   public String info(){
31   LOG.log(Level.INFO, "info is being called UserController showLogin");
32   return restTemplate.getForObject("http://localhost:8762/info",String.cl
ass);
33   }
34
35   @Autowired
36   private RestTemplate restTemplate;
37
38   @Bean
39   public RestTemplate getRestTemplate(){
40   return new RestTemplate();
41   }
42
43   @Bean
44   public Sampler defaultSampler() {
45   return Sampler.ALWAYS_SAMPLE;
46   }
47 }
```

2.4、启动运行

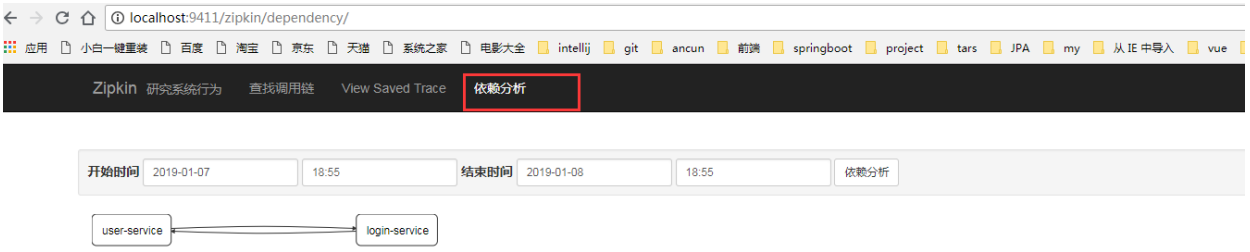
启动运行着三个项目，访问：http://localhost:8762/sayHello 效果如图：



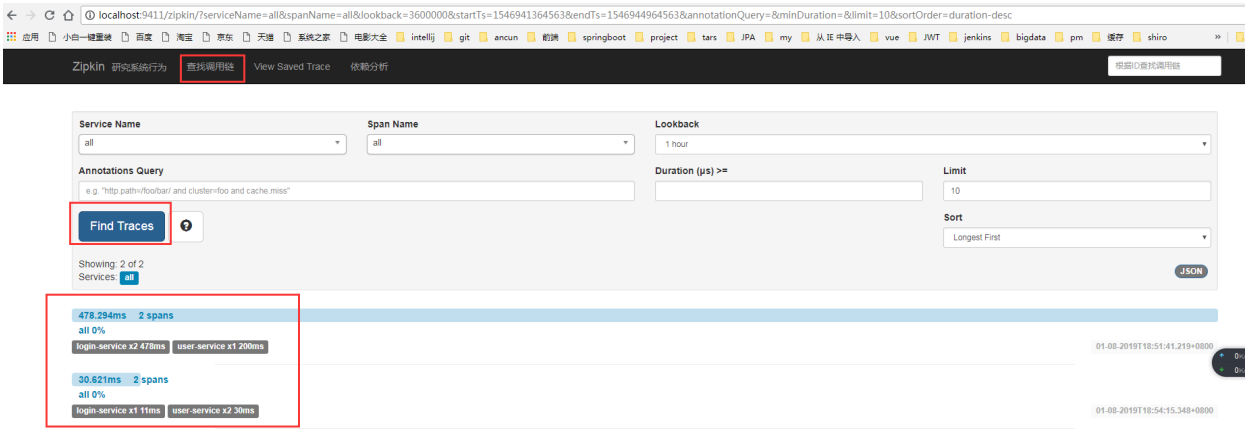
访问：http://localhost:10180/showLogin 效果如图：

然后再刷新一下：http://localhost:9411/zipkin/ 得到效果如下：

点击“依赖分析”，可以发现服务的依赖关系，如图：



点击“查找调用链”，可以看到具体服务相互调用的数据，如图：



----------忽略线----------

三、Zipkin-server创建

由于SpringCloud已经提供了ZipkinServer的运行版，所以这里把zipkin服务作为一个部分单独处理。

搭建zipkin服务端，并不是很难，搭建的步骤：1、配置pom.xml，添加zipkin的依赖包,2、配置application.yml,3、配置启动器，添加@EnableZipkinServer注解，开启服务端

（As of sleuth 2.x there are no extensions to zipkin server anymore. Starting with 1.3, for example, you can use standard zipkin for rabbit mq transport etc.）

由于springboot2.x以后，官方已经不建议自行搭建zipkin服务了，就连@EnableZipkinServer这个注解也被打上了@Deprecated，具体可去github上查看。故不再继续处理

3.1、添加依赖

pom.xml如下：

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>xyz.jiangnanke</groupId>
  <artifactId>zipkin-server</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>zipkin-server</name>
  <description>Demo project for Spring Boot</description>

  <!--<parent>-->
  <!--<groupId>xyz.jiangnanke</groupId>-->
  <!--<artifactId>main</artifactId>-->
  <!--<version>0.0.1-SNAPSHOT</version>-->
  <!--</parent>-->

  <repositories>
  <repository>
  <id>spring-milestones</id>
  <name>Spring Milestones</name>
  <url>https://repo.spring.io/milestone</url>
  <snapshots>
  <enabled>false</enabled>
  </snapshots>
  </repository>
  </repositories>

  <!-- 引用父类依赖 -->
  <parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>1.5.10.RELEASE</version>
  </parent>

  <properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
  <java.version>1.8</java.version>
```

```
</properties>
<dependencies>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>
<!--eureka server -->
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-starter-eureka-server</artifactId>
<version>1.3.0.RELEASE</version><!--$NO-MVN-MAN-VER$-->
<exclusions>
<exclusion>
<groupId>javax.servlet</groupId>
<artifactId>servlet-api</artifactId>
</exclusion>
</exclusions>
</dependency>

<!-- 导入zipkin 的依赖包 -->
<dependency>
<groupId>io.zipkin.java</groupId>
<artifactId>zipkin-server</artifactId>
<version>1.30.0</version>
</dependency>

<dependency>
<groupId>io.zipkin.java</groupId>
<artifactId>zipkin-autoconfigure-ui</artifactId>
<version>1.30.0</version>
</dependency>
<!--保存到数据库需要如下依赖-->
<dependency>
<groupId>io.zipkin.java</groupId>
<artifactId>zipkin-autoconfigure-storage-mysql</artifactId>
</dependency>
```

```
<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-jdbc</artifactId>
</dependency>

<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
</dependency>
</dependencies>
<build>
<plugins>
<!-- 添加spring的插件， 就可以直接通过 mvn spring-boot:run 运行了 -->
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
 <dependencies>
 <dependency>
 <groupId>org.springframework</groupId>
 <artifactId>springloaded</artifactId>
 <version>1.2.4.RELEASE</version>
 </dependency>
 </dependencies>
 </plugin>
 </plugins>
 </build>
</project>
```

3.2、配置环境

application.yml文件如下：

```
#配置当前服务的名称
spring:
 application:
 name: zipkin-server
 sleuth:
```

```
  enabled: false
  datasource:
  schema[0]: classpath:/zipkin.sql
  url: jdbc:mysql://localhost:3306/zipkin?autoReconnect=true&useUnicode=true&characterEncoding=UTF-8&zeroDateTimeBehavior=convertToNull&useSSL=false
   username: root
   password: root
   driver-class-name: com.mysql.jdbc.Driver
   continue-on-error: true

 server:
   port: 18080
 eureka:
   client:
   serviceUrl:
   defaultZone: http://localhost:8761/eureka/

 #zipkin:
 # storage:
 # type: mysql
```

3.3、配置启动器

启动类如下：

```
package xyz.jiangnanke.zipkinserver;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
import zipkin.server.EnableZipkinServer;

@SpringBootApplication
@EnableEurekaClient
 @EnableZipkinServer
 public class ZipkinServerApplication {

   public static void main(String[] args) {
   SpringApplication.run(ZipkinServerApplication.class, args);
   }

```

```
}

```

3.4、sql文件

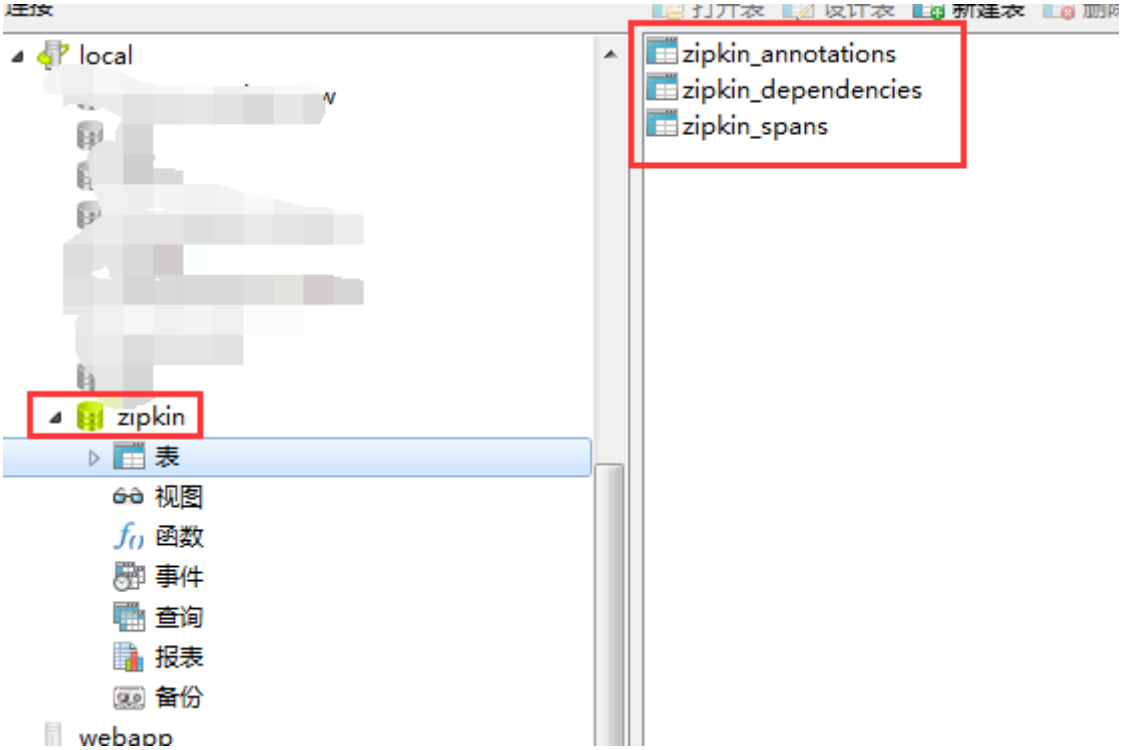
官网有sql文件https://github.com/openzipkin/zipkin/blob/master/zipkin-storage/mysql/src/main/resources/mysql.sql，zipkin.sql的内容如下：

```
/*
Navicat MySQL Data Transfer

Source Server : localhost
Source Server Version : 50718
Source Host : localhost:3306
Source Database : zipkin

Target Server Type : MYSQL
Target Server Version : 50718
File Encoding : 65001

Date: 2017-05-04 13:15:00
*/

SET FOREIGN_KEY_CHECKS=0;

-- ----------------------------
-- Table structure for zipkin_annotations
-- ----------------------------
CREATE TABLE `zipkin_annotations` (
  `trace_id` bigint(20) NOT NULL COMMENT 'coincides with zipkin_spans.trace_id',
  `span_id` bigint(20) NOT NULL COMMENT 'coincides with zipkin_spans.id',
  `a_key` varchar(255) NOT NULL COMMENT 'BinaryAnnotation.key or Annotation.value if type == -1',
  `a_value` blob COMMENT 'BinaryAnnotation.value(), which must be smaller than 64KB',
  `a_type` int(11) NOT NULL COMMENT 'BinaryAnnotation.type() or -1 if Annotation',
  `a_timestamp` bigint(20) DEFAULT NULL COMMENT 'Used to implement TTL; Annotation.timestamp or zipkin_spans.timestamp',
  `endpoint_ipv4` int(11) DEFAULT NULL COMMENT 'Null when Binary/Annotation.endpoint is null',
```

```
  `endpoint_ipv6` binary(16) DEFAULT NULL COMMENT 'Null when Binary/Annotation.endpoint is null, or no IPv6 address',
  `endpoint_port` smallint(6) DEFAULT NULL COMMENT 'Null when Binary/Annotation.endpoint is null',
  `endpoint_service_name` varchar(255) DEFAULT NULL COMMENT 'Null when Binary/Annotation.endpoint is null',
  UNIQUE KEY `trace_id` (`trace_id`,`span_id`,`a_key`,`a_timestamp`) COMMENT 'Ignore insert on duplicate',
  KEY `trace_id_2` (`trace_id`,`span_id`) COMMENT 'for joining with zipkin_spans',
  KEY `trace_id_3` (`trace_id`) COMMENT 'for getTraces/ByIds',
  KEY `endpoint_service_name` (`endpoint_service_name`) COMMENT 'for getTraces and getServiceNames',
  KEY `a_type` (`a_type`) COMMENT 'for getTraces',
  KEY `a_key` (`a_key`) COMMENT 'for getTraces'
) ENGINE=InnoDB DEFAULT CHARSET=utf8 ROW_FORMAT=COMPRESSED;

-- ----------------------------
-- Table structure for zipkin_dependencies
-- ----------------------------
CREATE TABLE `zipkin_dependencies` (
  `day` date NOT NULL,
  `parent` varchar(255) NOT NULL,
  `child` varchar(255) NOT NULL,
  `call_count` bigint(20) DEFAULT NULL,
  UNIQUE KEY `day` (`day`,`parent`,`child`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 ROW_FORMAT=COMPRESSED;

-- ----------------------------
-- Table structure for zipkin_spans
-- ----------------------------
CREATE TABLE `zipkin_spans` (
  `trace_id` bigint(20) NOT NULL,
  `id` bigint(20) NOT NULL,
  `name` varchar(255) NOT NULL,
  `parent_id` bigint(20) DEFAULT NULL,
  `debug` bit(1) DEFAULT NULL,
  `start_ts` bigint(20) DEFAULT NULL COMMENT 'Span.timestamp(): epoch micros used for endTs query and to implement TTL',
  `duration` bigint(20) DEFAULT NULL COMMENT 'Span.duration(): micros used for minDuration and maxDuration query',
  UNIQUE KEY `trace_id` (`trace_id`,`id`) COMMENT 'ignore insert on duplicate',
```

```
  KEY `trace_id_2` (`trace_id`,`id`) COMMENT 'for joining with zipkin_annotations',
  KEY `trace_id_3` (`trace_id`) COMMENT 'for getTracesByIds',
  KEY `name` (`name`) COMMENT 'for getTraces and getSpanNames',
  KEY `start_ts` (`start_ts`) COMMENT 'for getTraces ordering and range'
) ENGINE=InnoDB DEFAULT CHARSET=utf8 ROW_FORMAT=COMPRESSED;
```

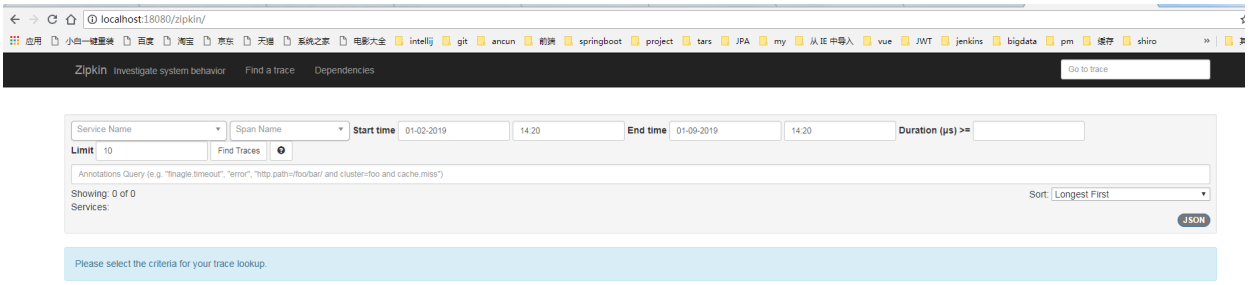
创建好的数据库（运行项目即可创建表）如图：



3.5、运行启动

访问：http://localhost:18080/zipkin/

结果如下：



（当前springboot2.X版本和zipkin1.x版本不兼容，所以改成springboot1.5版本的了，不建议自己建zipkin-server）

附录：

日志聚合工具（例如**Kibana，Splunk**等），则可以对发生的事件进行排序。

Logstash：

Logstash是一个具有实时流水线功能的开源数据收集引擎。 Logstash可以动态统一来自不同来源的数据，并将数据标准化为您选择的目的地。 为各种高级下游分析和可视化用例清理和民主化所有数据。

虽然Logstash最初推动了日志收集的创新，但其功能远远超出了该用例。 任何类型的事件都可以通过广泛的输入，过滤和输出插件进行丰富和转换，许多本机编解码器进一步简化了摄取过程。 Logstash通过利用更大容量和更多数据来加速您的见解。

Grok是Logstash过滤器的基础，可以无处不在地用于从非结构化数据中获取结构。 享受丰富的集成模式，旨在帮助快速解决Web，系统，网络和其他类型的事件格式。

Logback

参考资料：

https://springcloud.cc/

https://zipkin.io/

https://github.com/spring-cloud/spring-cloud-sleuth

https://blog.csdn.net/forezp/article/details/81041078

http://cloud.spring.io/spring-cloud-static/Finchley.RELEASE/single/spring-cloud.html

https://blog.csdn.net/zhengzizhi/article/details/81151408

https://blog.csdn.net/smartdt/article/details/79077110

https://blog.csdn.net/yelllowcong/article/details/79593579

https://blog.csdn.net/weixin_43633424/article/details/83829096

https://blog.csdn.net/niugang0920/article/details/81259990
https://blog.csdn.net/qq_27384769/article/details/79704059
https://blog.csdn.net/tudou201601/article/details/79123912