SUMK框架入门

[快速搭建web应用 2](#_Toc42526380)

[编写第一个web接口 3](#_Toc42526381)

[使用数据库 3](#_Toc42526382)

[使用SDB 4](#_Toc42526383)

[引入mybatis 5](#_Toc42526384)

[配置 5](#_Toc42526385)

[日志 5](#_Toc42526386)

[Web权限校验 6](#_Toc42526387)

[多接入类型支持 6](#_Toc42526388)

[单点登录限制 6](#_Toc42526389)

[Web服务安全性 6](#_Toc42526390)

[Redis以及分布式session 7](#_Toc42526391)

[统一配置中心 7](#_Toc42526392)

[缓存系统 7](#_Toc42526393)

[读写分离 8](#_Toc42526394)

[多数据源 8](#_Toc42526395)

[监听表的读写 8](#_Toc42526396)

[接口文档 8](#_Toc42526397)

[接口访问统计 9](#_Toc42526398)

[微服务的搭建 9](#_Toc42526399)

[编写第一个微服务 9](#_Toc42526400)

[调用微服务 9](#_Toc42526401)

[通过接口调用的微服务 10](#_Toc42526402)

[代码自动生成 11](#_Toc42526403)

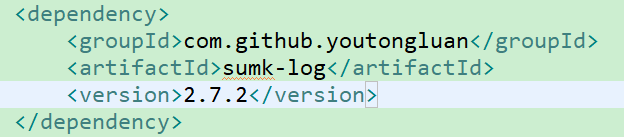
[材料下载地址 11](#_Toc42526404)

## 快速搭建web应用

1. 创建maven工程，并将工程的jdk版本设为8



1. 引入sumk.jar及其依赖。因为sumk-log依赖于sumk，所以引入sumk-log.jar就会把sumk.jar一起引入



1. 启动程序，关键代码是SumkServer.start()或SumkServer.main(args)，其它代码都是可选的
2. 配置信息，所有的配置文件都在app.properties文件中



完成以上四步，就搭建了一个web应用。

参见<https://github.com/youtongluan/sumk-server-demo/tree/master/web-server>

## 编写第一个web接口



在浏览器输入后面的网址，就可以访问了。[http://localhost:8081/rest/info?data={“age”:23}](http://localhost:8081/rest/info?data=%7b)

因为name没有声明是必传参数，所以上面的请求参数里可以没有name字段。如果要将name设为必传，需要用上@Param注解，然后访问地址要改成这样：http://localhost:8081/rest/info?data={"name":"张三","age":23}

为了安全性，客户端推荐使用POST请求，为了演示方便，这里都是用get请求

## 使用数据库

1. 在app.properties中添加数据库配置



1. 创建表的pojo对象。
   1. @SoftDelete表示软删除，如果使用物理删除，就可以去掉这个注解。
   2. lastUpdate对应的数据库字段名为last\_update，如果不是这个名字，要通过@Column来指定。



1. 在mysql上创建名字为sumk的数据库，数据库编码utf-8。建表可以通过语句，也可以通过CodeToolTest. generateDBTable()来自动创建表格。
2. 使用数据库，比如往数据库写入一条记录



在浏览器输入下面地址，就可以插入一条记录了，返回值是记录的id。http://localhost:8081/rest/insert?data={"name":"测试","age":23}

1. DB操作的文档参见这里：<https://my.oschina.net/u/819657/blog/3008795>
2. DB操作默认为mysql，如果使用progressSQL等其它数据库，请引入他们的驱动包，并修改s.db.sumk.1.driverClassName指定驱动类

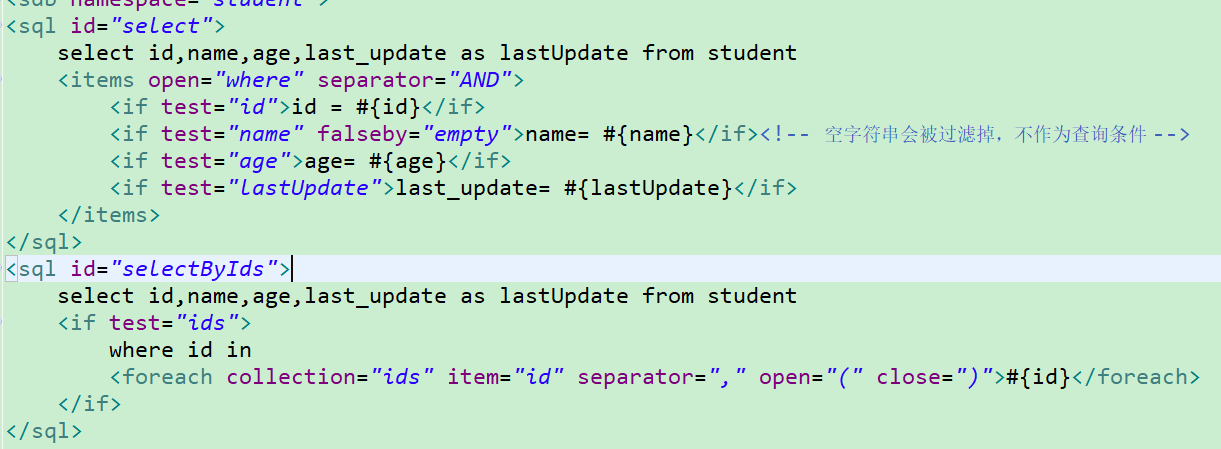
## 使用SDB

Sumk的orm虽然提供了缓存、事件等功能，但它只支持单表操作。如果需要做关联操作，推荐使用SDB进行操作。SDB类似于mybatis，相比于mybatis，SDB的功能要少很多，但是它更轻，占资源更少，并且无缝对接sumk-log。此外SDB还支持sql文件的动态变更并且支持远程sql文件（需要实现它的适配器）。SDB使用方法如下：

1. 在resources底下创建sql文件夹，然后在里面存放xml文件。



1. 编写xml文件



1. 通过SDB调用语句。SDB只支持map参数，并且map中的value只能是String、原始对象、集合。好在sumk提供了S.beans工具类做对象和map间的转换。



## 引入mybatis

1. 在pom.xml中引入mybatis.jar



1. 在resource的batis/sumk底下放置mybatis的sql文件



1. 使用SqlSessionHolder.session().selectOne(\*\*\*)就可以使用原生的mybatis方法进行操作

## 配置

每个应用都需要配置，sumk的配置默认放在app.properties文件中，属性支持**动态变更**，修改app.properties的值，一般1分钟内就能读到最新的值。这里除了存放框架所需要的配置，也可以存放开发者自定义的配置。框架配置以sumk、s这2个开头，开发者要避开这几个前缀。获取配置方式如下图所示：



## 日志

详情访问<https://github.com/youtongluan/sumk-log>，日志会额外打印用户id等，支持自定义日志，提供统一日志接口。

日志的几个有趣的应用：

1. 因为配置中心有定时更新的功能，所以日志的级别以及输出方式可以动态调整
2. 因为配置中心支持统一配置中心，所以可以通过统一配置统一修改各个应用的日志级别。如果统一配置中心有管理界面的话，就可以通过界面来调整日志属性
3. 可以通过自定义日志接口来自定义日志的记录行为，自定义方式：通过LogAppenderFactory. registeAppender()将你的LogAppender注册进来。然后在配置里使用s.log.名称=\*\*\*来配置它
4. 通过Plugin和统一日志的结合来做性能采集和预警。在Plugin里编写指标采集，比如JVM指标等，然后通过上述的自定义日志方式，将该模块的日志传输到远程的分析程序里。

## Web权限校验

1. 用户登录，继承AbstractLoginServlet并且添加@Bean注解，就会被当成用户登录接口（见MyLoginServlet），访问方式（返回值是密钥）：<http://localhost:8081/login?username=admin&password=123456&code=9999>
2. 需要登录后才能访问的接口，使用@Web(requireLogin=**true**)注解就行（见UserAction.userInfo()接口），访问方式：<http://localhost:8081/rest/userInfo>
3. 如果需要更复杂的权限校验方式，只要继承WebFilter类，并用@Bean注解就行了

## 多接入类型支持

随着业务发展，web服务除了提供给自己使用外，也可能要提供给第三方应用接入。它们使用的接口和业务逻辑可能都是完全不同的。比如个人用户、企业用户、代理商，他们的用户、session、权限应该都是完全不同的。通过指定@Web中有type属性以及重载AbstractSessionServlet的getType()方法，就可以实现上述功能。

## 单点登录限制

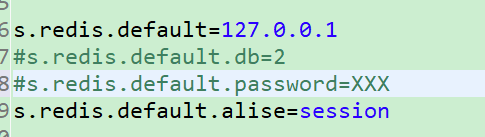
许多应用都有pc版、app版、微信版等多种版本，这样用户可能在多个地方同时登录，默认情况下多个地方登录同时有效，如果想限制用户只能在一个地方登录，及登录了app之后，pc上就要被强制退出。这时你只要加个配置就好了：sumk.http.session.single=1

## Web服务安全性

Web服务除了使用https外，sumk框架内置了签名验证、参数加密、返回值加密等，签名、加密的算法都可以自定义，参见@Web注解。以下地址可以找到例子<https://github.com/youtongluan/sumk/tree/master/src/test/java/org/test>

## Redis以及分布式session

随着业务量的增长，可能会引入nginx等负载均衡，这时候就需要分布式session。对于sumk框架，只需要加入如下配置，就启用了分布式缓存。其中session表示启用分布式缓存，default表示系统中引入了redis功能。



redis使用示例：



测试网址：[http://localhost:8081/rest/incrInRedis?data={"name":"test"}](http://localhost:8081/rest/incrInRedis?data=%7b%22name%22:%22test%22%7d)

## 统一配置中心

一旦应用部署了多个，就会需要配置中心。Sumk内置了http和zookeeper两种协议的配置中心。Spring boot的配置中心是http协议，sumk能够兼容它。如果这2种不满足需求，开发者还可以使用自定义的配置管理，比如本地文件+配置中心的方式

使用示例：

UrlSystemConfig config=new UrlSystemConfig(new URL("http://localhost:8080/app.properties"));

SumkServer.start(config);

## 缓存系统

ORM默认使用了表级缓存，不需要开发者关心。有三种方式可以关闭缓存。

* 配置中的sumk.sql.fromCache和sumk.sql.toCache是系统级配置
* @Table中的cacheType是表级配置
* Insert、Select等类的fromCache()、ToCache()是方法级开关

## 读写分离

分布式session只是web端的扩展，随着业务发展，数据库也要扩展。读写分离是很常见的读写分离，它的好处是不破坏数据库事务。在sumk中，只要多配置几个数据源，示例中的数据源只配置了s.db.sumk.1，我们可以加上s.db.sumk.2等。并且可以通过weight配置它们的权重。

## 多数据源

示例中的数据源只配置了s.db.sumk.1，如果我们增加s.db.test.1的配置，就增加了一个test数据源，它跟sumk没有任何关系。使用的时候，通过@Box(dbName=”test”)就可以使用这个数据源。

## 监听表的读写



实现DBListener接口的bean，就能监听到数据库变更。上面的例子是监听表的insert行为

## 接口文档

1. 随机使用一串字符，比如helloworld，然后用md5工具生成它的哈希值，示例代码：S.hash. digest ("helloworld".getBytes())，它等到值fc5e038d38a57032085441e7fe7010b0
2. 在配置文件中增加：sumk.acts.md5=fc5e038d38a57032085441e7fe7010b0
3. 使用<http://localhost:8081/_sumk_acts?sign=helloworld&mode=http&pretty=1>可以查看web的接口出入参，开发者可以在这个基础上，制作文档生成工具。使用<http://localhost:8081/_sumk_acts?sign=helloworld&mode=rpc&pretty=1>，可以查看rpc接口出入参情况。

## 接口访问统计

1. 随机使用一串字符，比如helloworld，然后用md5工具生成它的哈希值，示例代码：S.MD5.encrypt("helloworld".getBytes())，它等到值fc5e038d38a57032085441e7fe7010b0
2. 在配置文件中增加：sumk.http.monitor=fc5e038d38a57032085441e7fe7010b0
3. 访问一些web接口，输入后面的地址，就可以查看地址的有效访问情况：<http://localhost:8080/_sumk_monitor?sign=helloworld&statis=1>

## 微服务的搭建

1. 创建一个maven工程，引入sumk.jar及其依赖包（推荐通过sumk-log来引入sumk）。
2. 编写启动类，这个步骤也与web服务的搭建一样（实际上一个工程既可以提供web服务、也可以提供微服务）
3. 在app.properties中做如下配置



这样就完成了微服务工程的搭建。因为微服务依赖于zookeeper，所以要先启动zookeeper才能启动微服务程序。示例用的是本地zookeeper。工程示例：<https://github.com/youtongluan/sumk-server-demo/tree/master/rpc-server>

## 编写第一个微服务



只要在类上加@Bean注解，在方法上加@Soa注解就可以了。前提是该类的包名在sumk.ioc配置的目录下

## 调用微服务

Sumk的微服务调用支持同步、异步、回调三种方式。其中回调可以同时与前2种存在。增加使用的灵活性。支持数组、map、json三种传参方式，相比之下数组使用起来最方便，推荐使用数组方式。

1. 在web-server工程里，添加下面的配置



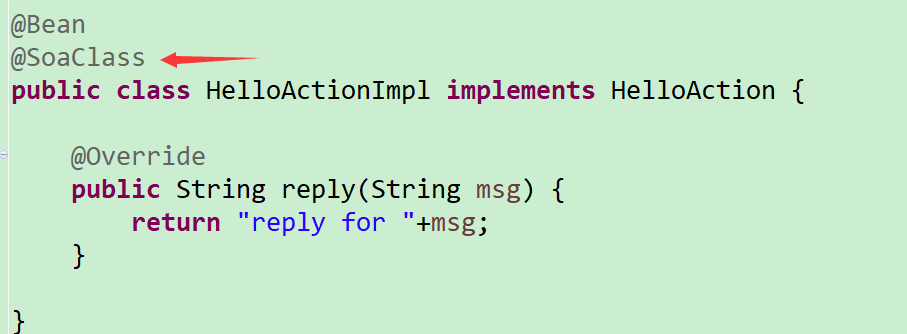
1. 调用echo服务



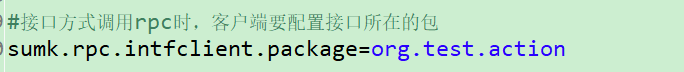
在浏览器里输入http://localhost:8081/rest/echoFromRpc?data={"name":"游夏"} ，就可以调用echoFromRpc方法，然后通过它来调用微服务。

## 通过接口调用的微服务

1、服务端需要在实现类上加@SoaClass注解。实现类如果没有继承接口，或继承了多个接口，就需要在被调用方建一个同包名、通类名的接口（从class变成了interface）



2、调用方配置要扫描接口所在的包名，该包及子包下的所有接口都被认为是rpc接口。也可以用sumk.rpc.intfclient.interface来指定具体的接口。如果要排除掉里面的某些接口，用sumk.rpc.intfclient.exclude来指定，它支持在头尾地方出现通配符\*。



1. 发起rpc调用



## 代码自动生成

1. 在maven上引入sumk-codetool,引入方式如下：
2. 在配置文件中配置sumk.code.output=D:\\output\\ ，值是生成文件的存放目录。
3. 执行CodeTool.generateDao(Student.class); 就能生成dao代码，其中Student是经过@Table注解的pojo类

更详细的例子参见：<https://github.com/youtongluan/sumk-codetool/tree/master/sumk-codetool-test>

## 材料下载地址

上述的工程示例地址：<https://github.com/youtongluan/sumk-server-demo>

zookeeper下载地址：<https://www.apache.org/dyn/closer.cgi/zookeeper/>

mysql和redis因不同操作系统而不同，需要大家自己下载

## 常用配置

|  |  |  |
| --- | --- | --- |
| 配置名 | 作用 | 默认自 |
| sumk.appId | 应用的AppId |  |
| sumk.ioc | Ioc的包名，多个以逗号隔开 |  |
| sumk.log.level | 日志级别 | Info |
| sumk.log.body.maxlength | 日志体的最大长度 | 1500 |
| sumk.log.maxLogNameLength | 日志名称的最大长度 | 32 |
| sumk.log.console | 1表示无论如何都打印控制台日志 | 0 |
| http服务配置 |  |  |
| sumk.http.port | http服务的端口，大于0才有意义 | -1 |
| sumk.jetty.web.root | Web项目的项目名 | / |
| sumk.http.pattern | @Web所在类的路径，支持通配符\* | \* |
| sumk.jetty.resource | Web项目静态资源的目录 |  |
| sumk.http.log.reqsize | http日志中请求的最大长度 | 1000 |
| sumk.http.log.respsize | http日志中响应的最大长度 | 1000 |
| sumk.http.log.warn.time | http请求的处理超过这个时间，http的日志级别变为warn | 3000 |
| sumk.http.log.info.time | http请求的处理超过这个时间，http的日志级别变为info | 1000 |
| sumk.http.body.maxLength | http请求body的最大长度，单位byte | 100M |
| sumk.jetty.ssl.keyStore  sumk.jetty.ssl.storePassword  sumk.jetty.ssl.managerPassword  sumk.jetty.ssl.alias | https的相关配置 |  |
| sumk.http.session.single | 1启用同账号的互踢功能 | 0 |
| sumk.http.session.timeout | http的Session过期时间，单位秒 | 1800 |
| sumk.http.error.XXX | 定制某个异常码的出错信息 |  |
| 微服务的配置 |  |  |
| sumk.rpc.port |  |  |
| sumk.zkurl | 微服务用到的zk的地址 |  |
| sumk.rpc.pattern | @Soa所在类的路径，支持通配符\* | \* |
| sumk.rpc.client.start | 是否初始化rpc客户端 | 0 |
| sumk.rpc.server.register | 如果为0，就不注册zookeeper | 1 |
| sumk.rpc.weight | 当前微服务服务器的权重 | 5 |
| Redis配置 | \*与default是等价的，表示默认redis，如果是某个名称的实例，将\*改为redis名 |  |
| s.redis.\* | 默认Redis实例的地址 |  |
| s.redis.\*.maxAttempts | 默认redis实例的最大尝试次数 | 3 |
| s.redis.\*.db | 默认redis实例使用的db库 | 0 |
| s.redis.\*.alias | 默认redis实例的别名，别名可以在RedisPool.get()中使用 |  |
| s.redis.\*.password | 默认redis实例的密码 |  |
| s.redis.\*.timeout | 默认redis实例的超时时间 |  |
| s.redis.\*.type | Redis的类型：0普通、1 cluster、 2 sentinel | 0 |
| 数据库配置 | 配置中的sumk是数据源名称，就是@Box中的value,1是该数据源的第几个数据库实例 | 无论读库还是写库，都可以配置多个 |
| s.db.sumk.1.driverClassName | 驱动类型 | com.mysql.jdbc.Driver |
| s.db. sumk.1.url |  |  |
| s.db. sumk.1.username |  |  |
| s.db. sumk.1.password |  |  |
| s.db. sumk.1.weight | 数据库的权重 | 1 |
| s.db. sumk.1.maxTotal |  |  |
| s.db. sumk.1.maxIdle |  |  |
| s.db. sumk.1.minIdle |  |  |
| s.db. sumk.1.type | 数据库的读写类型。wr是读写,read是只读,write是只写 | wr |
| sumk.db.password.encry | 1表示启用数据库密码的加密模式。使用项目中的《数据库密码加密.exe》对密码进行加密 | 0 |
|  |  |  |