**目录**

[前言 1](#_Toc21968)

[1服务端配置： 1](#_Toc21235)

[2服务服务开发： 3](#_Toc12483)

[2.1 定义类(class) 3](#_Toc25304)

[2.2 定义方法(method) 4](#_Toc18555)

[2.2.1 细节： 4](#_Toc18590)

[3 WEB 后台管理页面： 5](#_Toc1794)

[3.1 访问限制 5](#_Toc22943)

[4客户端开发： 6](#_Toc6620)

# 前言

本starter适用于spring-boot 1.5.6.RELEASE 及以上版本：

基于netty-all 4.1.43.Final 实现

Git地址：<http://47.102.201.6:9902/marsatg/poseidon-spring-boot-starter.git>

Maven执行install命令后，将项目安装到本地maven 仓库中

在需要引用本工程的pom文件中加入如下配置：

<**dependency**>  
 <**groupId**>org.marsatg</**groupId**>  
 <**artifactId**>poseidon-spring-boot-starter</**artifactId**>  
 <**version**>1.0</**version**>  
</**dependency**>

# 1服务端配置：

作为服务端使用时：

（注：本文档所有非指明包的注解均在 org.marsatg.annotation包下）

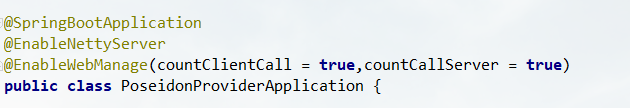
服务端注解：

@EnableNettyServer

开启Web后台页面注解：

@EnableWebManage(countClientCall = **true**,countCallServer = **true**)

其中 countClientCall = **true**,countCallServer = **true** 分别表示作为服务端时启用记录客户端的调用次数统计功能，和作为客户端时启用调用服务端的次数统计功能

如图所示：

**application.properties** 中必须指明当前服务的服务名称和tcp端口

#服务端主机名称（必填）

**netty-local.applicationName**=**nettyServer**

#服务端主机绑定TCP端口（必填） **netty-local.applicationPort**=**12345**

服务端默认使用CachedThreadPool 处理并发请求

可以使用以下配置指定为FixedThreadPool，并制定最大线程数量控制并发

#使用FixedThreadPool线程池

**netty-local.isFixedThreadPool**=**true**

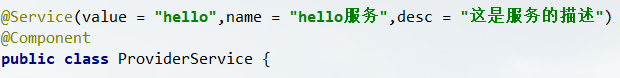
#使用FixedThreadPool线程池最大线程数量

**netty-local.threads**=**6**

# 2服务端开发：

定义项目需要用到的服务：

## 2.1 定义类(class)



其中@Component注解为spring内置注解：

org.springframework.stereotype.Component

该注解标识本服务将被spring的实例化已存放至IOC容器中

@Service 注解有三个参数：

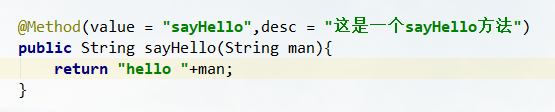
(value = **"hello"**,name = **"hello服务"**,desc = **"这是服务的描述")**

value = **"hello"** 客户端调用时指定的服务名称，

name = **"hello服务"** web后台页面展示的服务名称，

desc = **"这是服务的描述"** web后台页面展示的描述

## 2.2 定义方法(method)



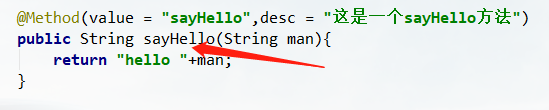
注解 @Method(value = **"sayHello"**,desc = **"这是一个sayHello方法"**)

包含两个参数：

value = **"sayHello"** 客户端调用时指定的方法名称

desc = **"这是一个sayHello方法"** web后台页面展示的方法描述

注意：value 属性和desc 属性可以不写，当没有value属性时，将使用具体的方法名作为调用名称



当方法名被修改后，客户端将找不到方法调用，建议使用value标注

### 2.2.1 细节：

同一个class中，不能出现两个一模一样的 @Method 注解标注的方法

(参数value一样)

类似于同一个Controller中，不能出现两个一样的RequestMapping

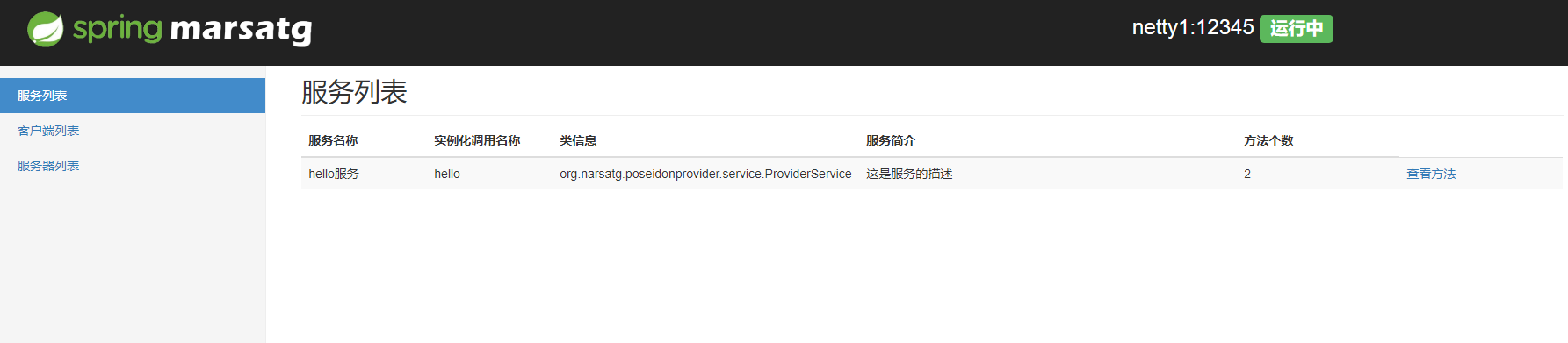
# 3 WEB 后台管理页面：

访问地址 http://{ip}:{port}/{contextPath}/marsatg.html

本例中没有配置 contextPath 因此访问地址为：

<http://localhost:8081/marsatg.html>

如图:



## 3.1 访问限制

如果页面无法访问:请保证服务是否有过滤器对静态页面进行了拦截：

保证过滤器对包含 **marsatg .js .css**  关键字的请求予以开放

如果**application.properties** 中有以下配置项

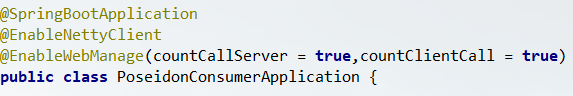
**spring.resources.static-locations** 配置项

则需要添加以下路径

**spring.resources.static-locations=classpath:/resources/static**

如果未配置**spring.resources.static-locations**配置,则不需要上述配置

# 4客户端开发：



客户端需要开启 @EnableNettyClient 注解

**application.properties** 必须配置以下信息

#当前客户端名称（必填）

**netty-local.applicationName**=**consumer001**

#服务端主机IP（必填）

**netty-local.server[0].host**=**localhost**

#服务端主机调用名称（必填，客户调用将基于此服务器名称） **netty-local.server[0].name**=**nettyServer**

#服务端主机调用端口（必填） **netty-local.server[0].port**=**12345**

#服务端主机断线重连间隔时间/毫秒（非必填，默认5000毫秒） **netty-local.server[0].reConnectTnterval**=**5000**

#服务端主机断线重连间隔最大次数（非必填，默认12次） **netty-local.server[0].maxReConnectTimes**=**12**

#客户端最大请求超时时间/毫秒，默认6000毫秒，（非必填） **netty-local.requestBlockTimeout**=**5000**

客户端可配置多个server，例如：**netty-local.server[**下标**].xxx**

netty模式下，调用服务端如下：

1 (rest模式，推荐使用):

NettyRequest request = **new** NettyRequest(5000);  
request.invokeUrlBlockTimeout(NettyConstants.***RealnameAPI***,**"/project-service/delete-project"**,loginUser,id);

5000 ： ***本次请求最大的超时时间***

NettyConstants.***RealnameAPI : 服务端名称***

**"/project-service/delete-project" ： *rest风格url***

loginUser,id ： ***调用所需要的参数数组,需对应服务端的入口参数***

2 直接使用service和method调用（不推荐）

@Deprecated

**NettyRequest nettyRequest = NettyRequest.invokeBlock("nettyServer", "hello", "sayHello", "marsatg");**

**四个参数依次为：**

1服务器调用名称 上述 **netty-local.server[0].name**

2 服务器提供的service类调用名 上述：@Service 的value属性

3 服务器提供的method调用名 上述：@Method的value属性

4 @Method 指定的方法的参数数组 ([本例为java.lang.String])

org.marsatg.netty.**NettyRequest**类提供了两个静态方法

1 ***invokeBlock***阻塞调用，阻塞到服务器返回响应，或者阻塞到最大超时时间为止

2 ***invokeNonBlock***非阻塞调用，服务器不会返回响应，请求端执行完，自动结束

***invokeBlock*** 方法会执行完毕会返回 **NettyRequest** 类的示例化对象，可通过：

**Object result = nettyRequest.getResult();**

其中 result 即为服务器的响应结果