**netty-websocket-spring-boot-starter**

[English Docs](https://github.com/YeautyYE/netty-websocket-spring-boot-starter/blob/master/README.md)

**简介**

本项目帮助你在spring-boot中使用Netty来开发WebSocket服务器，并像spring-websocket的注解开发一样简单

**要求**

* jdk版本为1.8或1.8+

**快速开始**

* 添加依赖:

<dependency>

<groupId>org.yeauty</groupId>

<artifactId>netty-websocket-spring-boot-starter</artifactId>

<version>0.7.1</version>

</dependency>

* new一个ServerEndpointExporter对象，交给Spring容器，表示要开启WebSocket功能，样例如下:

@Configuration

**public** **class** **WebSocketConfig** {

@Bean

**public** ServerEndpointExporter **serverEndpointExporter**() {

**return** **new** **ServerEndpointExporter**();

}

}

* 在端点类上加上@ServerEndpoint、@Component注解，并在相应的方法上加上@OnOpen、@OnClose、@OnError、@OnMessage、@OnBinary、OnEvent注解，样例如下：

@ServerEndpoint

@Component

**public** **class** **MyWebSocket** {

@OnOpen

**public** **void** **onOpen**(Session session, HttpHeaders headers, ParameterMap parameterMap) **throws** IOException {

System.out.println("new connection");

String paramValue = parameterMap.getParameter("paramKey");

System.out.println(paramValue);

}

@OnClose

**public** **void** **onClose**(Session session) **throws** IOException {

System.out.println("one connection closed");

}

@OnError

**public** **void** **onError**(Session session, Throwable throwable) {

throwable.printStackTrace();

}

@OnMessage

**public** **void** **onMessage**(Session session, String message) {

System.out.println(message);

session.sendText("Hello Netty!");

}

@OnBinary

**public** **void** **onBinary**(Session session, **byte**[] bytes) {

**for** (**byte** b : bytes) {

System.out.println(b);

}

session.sendBinary(bytes);

}

@OnEvent

**public** **void** **onEvent**(Session session, Object evt) {

**if** (evt **instanceof** IdleStateEvent) {

IdleStateEvent idleStateEvent = (IdleStateEvent) evt;

**switch** (idleStateEvent.state()) {

**case** READER\_IDLE:

System.out.println("read idle");

**break**;

**case** WRITER\_IDLE:

System.out.println("write idle");

**break**;

**case** ALL\_IDLE:

System.out.println("all idle");

**break**;

**default**:

**break**;

}

}

}

}

* 打开WebSocket客户端，连接到ws://127.0.0.1:80

**注解**

**@ServerEndpoint**

当ServerEndpointExporter类通过Spring配置进行声明并被使用，它将会去扫描带有@ServerEndpoint注解的类 被注解的类将被注册成为一个WebSocket端点 所有的[配置项](https://gitee.com/Yeauty/netty-websocket-spring-boot-starter#%E9%85%8D%E7%BD%AE)都在这个注解的属性中 ( 如:@ServerEndpoint("/ws") )

**@OnOpen**

当有新的WebSocket连接进入时，对该方法进行回调 注入参数的类型:Session、HttpHeaders、ParameterMap

| **属性** | **默认值** | **说明** |
| --- | --- | --- |
| path | "/" | WebSocket的path,也可以用value来设置 |
| host | "0.0.0.0" | WebSocket的host,"0.0.0.0"即是所有本地地址 |
| port | 80 | WebSocket绑定端口号。如果为0，则使用随机端口(端口获取可见 [多端点服务](https://gitee.com/Yeauty/netty-websocket-spring-boot-starter#%E5%A4%9A%E7%AB%AF%E7%82%B9%E6%9C%8D%E5%8A%A1)) |
| bossLoopGroupThreads | 0 | bossEventLoopGroup的线程数 |
| workerLoopGroupThreads | 0 | workerEventLoopGroup的线程数 |
| useCompressionHandler | false | 是否添加WebSocketServerCompressionHandler到pipeline |
| prefix | "" | 当不为空时，即是使用application.properties进行配置，详情在 [通过application.properties进行配置](https://gitee.com/Yeauty/netty-websocket-spring-boot-starter#%E9%80%9A%E8%BF%87APPLICATION.PROPERTIES%E8%BF%9B%E8%A1%8C%E9%85%8D%E7%BD%AE) |
| optionConnectTimeoutMillis | 30000 | 与Netty的ChannelOption.CONNECT\_TIMEOUT\_MILLIS一致 |
| optionSoBacklog | 128 | 与Netty的ChannelOption.SO\_BACKLOG一致 |
| childOptionWriteSpinCount | 16 | 与Netty的ChannelOption.WRITE\_SPIN\_COUNT一致 |
| childOptionWriteBufferHighWaterMark | 64\*1024 | 与Netty的ChannelOption.WRITE\_BUFFER\_HIGH\_WATER\_MARK一致,但实际上是使用ChannelOption.WRITE\_BUFFER\_WATER\_MARK |
| childOptionWriteBufferLowWaterMark | 32\*1024 | 与Netty的ChannelOption.WRITE\_BUFFER\_LOW\_WATER\_MARK一致,但实际上是使用 ChannelOption.WRITE\_BUFFER\_WATER\_MARK |
| childOptionSoRcvbuf | -1(即未设置) | 与Netty的ChannelOption.SO\_RCVBUF一致 |
| childOptionSoSndbuf | -1(即未设置) | 与Netty的ChannelOption.SO\_SNDBUF一致 |
| childOptionTcpNodelay | true | 与Netty的ChannelOption.TCP\_NODELAY一致 |
| childOptionSoKeepalive | false | 与Netty的ChannelOption.SO\_KEEPALIVE一致 |
| childOptionSoLinger | -1 | 与Netty的ChannelOption.SO\_LINGER一致 |
| childOptionAllowHalfClosure | false | 与Netty的ChannelOption.ALLOW\_HALF\_CLOSURE一致 |
| readerIdleTimeSeconds | 0 | 与IdleStateHandler中的readerIdleTimeSeconds一致，并且当它不为0时，将在pipeline中添加IdleStateHandler |
| writerIdleTimeSeconds | 0 | 与IdleStateHandler中的writerIdleTimeSeconds一致，并且当它不为0时，将在pipeline中添加IdleStateHandler |
| allIdleTimeSeconds | 0 | 与IdleStateHandler中的allIdleTimeSeconds一致，并且当它不为0时，将在pipeline中添加IdleStateHandler |

**@OnClose**

当有WebSocket连接关闭时，对该方法进行回调 注入参数的类型:Session

**@OnError**

当有WebSocket抛出异常时，对该方法进行回调 注入参数的类型:Session、Throwable

**@OnMessage**

当接收到字符串消息时，对该方法进行回调 注入参数的类型:Session、String

**@OnBinary**

当接收到二进制消息时，对该方法进行回调 注入参数的类型:Session、byte[]

**@OnEvent**

当接收到Netty的事件时，对该方法进行回调 注入参数的类型:Session、Object

**配置**

所有的配置项都在这个注解的属性中

**通过application.properties进行配置**

对注解中的prefix进行设置后，即可在application.properties中进行配置。如下：

* 首先在ServerEndpoint注解中设置prefix的值

@ServerEndpoint(prefix = "netty-websocket")

@Component

**public** **class** **MyWebSocket** {

...

}

* 接下来即可在application.properties中配置

netty-websocket.host=0.0.0.0

netty-websocket.path=/

netty-websocket.port=80

| **注解中的属性** | **配置文件中的key** | **例子** |
| --- | --- | --- |
| path | {prefix}.path | netty-websocket.path |
| host | {prefix}.host | netty-websocket.host |
| port | {prefix}.port | netty-websocket.port |
| bossLoopGroupThreads | {prefix}.boss-loop-group-threads | netty-websocket.boss-loop-group-threads |
| workerLoopGroupThreads | {prefix}.worker-loop-group-threads | netty-websocket.worker-loop-group-threads |
| useCompressionHandler | {prefix}.use-compression-handler | netty-websocket.use-compression-handler |
| optionConnectTimeoutMillis | {prefix}.option.connect-timeout-millis | netty-websocket.option.connect-timeout-millis |
| optionSoBacklog | {prefix}.option.so-backlog | netty-websocket.option.so-backlog |
| childOptionWriteSpinCount | {prefix}.child-option.write-spin-count | netty-websocket.child-option.write-spin-count |
| childOptionWriteBufferHighWaterMark | {prefix}.child-option.write-buffer-high-water-mark | netty-websocket.child-option.write-buffer-high-water-mark |
| childOptionWriteBufferLowWaterMark | {prefix}.child-option.write-buffer-low-water-mark | netty-websocket.child-option.write-buffer-low-water-mark |
| childOptionSoRcvbuf | {prefix}.child-option.so-rcvbuf | netty-websocket.child-option.so-rcvbuf |
| childOptionSoSndbuf | {prefix}.child-option.so-sndbuf | netty-websocket.child-option.so-sndbuf |
| childOptionTcpNodelay | {prefix}.child-option.tcp-nodelay | netty-websocket.child-option.tcp-nodelay |
| childOptionSoKeepalive | {prefix}.child-option.so-keepalive | netty-websocket.child-option.so-keepalive |
| childOptionSoLinger | {prefix}.child-option.so-linger | netty-websocket.child-option.so-linger |
| childOptionAllowHalfClosure | {prefix}.child-option.allow-half-closure | netty-websocket.child-option.allow-half-closure |
| readerIdleTimeSeconds | {prefix}.reader-idle-time-seconds | netty-websocket.reader-idle-time-seconds |
| writerIdleTimeSeconds | {prefix}.writer-idle-time-seconds | netty-websocket.writer-idle-time-seconds |
| allIdleTimeSeconds | {prefix}.all-idle-time-seconds | netty-websocket.all-idle-time-seconds |

application.properties中的key与注解@ServerEndpoint中属性的对应关系如下:

**自定义Favicon**

配置favicon的方式与spring-boot中完全一致。只需将favicon.ico文件放到classpath的根目录下即可。如下:

src/

+- main/

+- java/

| + <source code>

+- resources/

+- favicon.ico

**自定义错误页面**

配置自定义错误页面的方式与spring-boot中完全一致。你可以添加一个 /public/error 目录，错误页面将会是该目录下的静态页面，错误页面的文件名必须是准确的错误状态或者是一串掩码,如下：

src/

+- main/

+- java/

| + <source code>

+- resources/

+- public/

+- error/

| +- 404.html

| +- 5xx.html

+- <other public assets>

**多端点服务**

* 在[快速启动](https://gitee.com/Yeauty/netty-websocket-spring-boot-starter#%E5%BF%AB%E9%80%9F%E5%BC%80%E5%A7%8B)的基础上，在多个需要成为端点的类上使用@ServerEndpoint、@Component注解即可
* 可通过ServerEndpointExporter.getInetSocketAddressSet()获取所有端点的地址
* 当地址不同时(即host不同或port不同)，使用不同的ServerBootstrap实例
* 当地址相同,路径(path)不同时,使用同一个ServerBootstrap实例
* 当多个端点服务的port为0时，将使用同一个随机的端口号
* 当多个端点的port和path相同时，host不能设为"0.0.0.0"，因为"0.0.0.0"意味着绑定所有的host