迷恋着你微笑的人zz Lv2
2018年10月12日 阅读 867

关注

netty学习（三）springboot+netty+mybatis

[TOC]

前言

前面写到了springboot整合netty，只是使用netty将数据处理了，还没有存入数据库，现在就把mybatis加进去吧；刚好最近也有一位读者问到了这个问题；

正文

首先引入pom依赖

```java
<dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <!-- https://mvnrepository.com/artifact/io.netty/netty-all -->
        <dependency>
            <groupId>io.netty</groupId>
            <artifactId>netty-all</artifactId>
            <version>4.1.29.Final</version>
        </dependency>
        <!--mybatis-->
        <dependency>
            <groupId>org.mybatis.spring.boot</groupId>
            <artifactId>mybatis-spring-boot-starter</artifactId>
            <version>1.3.2</version>
        </dependency>
```

```
            <artifactId>mysql-connector-java</artifactId>
        </dependency>
        <!-- druid的starter -->
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>druid-spring-boot-starter</artifactId>
            <version>1.1.9</version>
        </dependency>
```

配置

配置文件配置：

```java
spring:
  datasource:
    druid:
      url: jdbc:mysql://localhost:3306/authority?useUnicode=true&characterEncoding=utf-8&us
      username: root
      password: 123456
      driver-class-name: com.mysql.jdbc.Driver
      initial-size: 2
      max-active: 30
      min-idle: 2
      max-wait: 1234
      pool-prepared-statements: true
      max-pool-prepared-statement-per-connection-size: 5
#mybatis
mybatis:
  mapper-locations: classpath:mapper/*.xml
  type-aliases-package: com.example.demo.entity
```

记住springboot的启动类上要扫描mapper接口包

springboot中启动netty

其实springboot中启动netty的方式有很多种，前一篇netty文章中我通过实现 `CommandLineRunner` 接口来启动netty服务，后面写了一篇springbean生命周期，发现 `springboot` 可以用配置bean的方式启动netty服务,在启动方法上添加 `@PostConstruct` 注解，代码如下：

```
@Component
public class NettyStart {
    @Resource
    private ServerHandler serverHandler;

    private EventLoopGroup bossGroup = new NioEventLoopGroup();
    private EventLoopGroup workGroup = new NioEventLoopGroup();

    /**
     * 启动netty服务
     * @throws InterruptedException
     */
    @PostConstruct
    public void start() throws InterruptedException {
        ServerBootstrap b=new ServerBootstrap();
        b.group(bossGroup,workGroup)
                .channel(NioServerSocketChannel.class)
                .option(ChannelOption.SO_BACKLOG,128)
                .childHandler(new ChannelInitializer<SocketChannel>() {
                    @Override
                    protected void initChannel(SocketChannel socketChannel) throws Exceptio
                        socketChannel.pipeline().addLast(serverHandler);
                    }
                });
        ChannelFuture future = b.bind(8080).sync();
        if (future.isSuccess()) {
            System.out.println("启动 Netty 成功");
        }
    }
    /**
     * 销毁
     */
    @PreDestroy
    public void destroy() {
        bossGroup.shutdownGracefully().syncUninterruptibly();
        workGroup.shutdownGracefully().syncUninterruptibly();
        System.out.println("关闭 Netty 成功");
    }
}
```

其中的 serverHandler 是通过依赖注入的方式注入的，这是为了后面 serverHandler 类中可以直接依赖注入；

```
@Component
@Sharable
public class ServerHandler extends ChannelInboundHandlerAdapter {
    @Resource
    private UserService userService;

    /**
     * 获取数据
     * @param ctx 上下文
     * @param msg 获取的数据
     */
    @Override
    public void channelRead(ChannelHandlerContext ctx, Object msg){
        ByteBuf readMessage= (ByteBuf) msg;
        System.out.println(readMessage.toString(CharsetUtil.UTF_8));
        List<User> users = userService.selectAll();
        users.forEach(user-> System.out.println(user.toString()));
    }
}
```

这个类加了 `@Component` 和 `@Sharable` 注解，前一个不用解释了，后一个个人觉得：这个类被spring托管了，那么这个类就是单例，相当于这个类是共享的，所以得加 `@Sharable` 注解，关于该注解，可以参考这篇文章：blog.csdn.net/supper10090... 这里我注入了 `UserService` 类，这里是操作数据库的方法

UserService 类

java

```
public interface UserService {
    /**
     * 获取所有数据
     * @return
     */
    List<User> selectAll();
}
```

UserServiceImpl 类

java

```
@Service
public class UserServiceImpl implements UserService {
    @Resource
```

```
        return userMapper.selectAll();
    }
}
```

实体类

java

```
public class User implements Serializable {
    private static final long serialVersionUID = -7760174501074818l7L;
    private int id;
    private String name;
    private String password;
    private int age;
    private Date birthday;
}
```

set,get方法省略

mapper接口

java

```
@Repository
public interface UserMapper {
    List<User> selectAll();
}
```

mapper的xml文件就补贴上来了;

输出结果

java

```
测试
User{id=1, name='admin', password='admin', age=10, birthday=Tue May 30 00:00:00 CST 2017}
User{id=2, name='teacher', password='teacher', age=30, birthday=null}
User{id=3, name='student', password='student', age=20, birthday=null}
```

我的mapper中只是查询数据库操作，并没有写如数据，如果要写入，修改mapper就行了；

在很多关于netty的项目或博客中都是通过new的方式添加 `pipeline` 的，而我这里用了注入的方式；我是因为我所做的这个项目是单机版的，相当于就是所有的功能都在一个项目里面，而且单个功能操作数据库的频率比较高，而且我做的这个项目处理不了高并发，并发高了就会gg；所以我所写的这些文章都是从项目上所学来的； 如果通过new的方式添加 `pipeline` ，那么在处理数据的方法中可以通过一个类实现 `ApplicationContextAware` 接口来获取spring上下文；然后在调用bean，操作数据库；下面是参考链接： 在非spring管理的类中,使用spring管理的类

- springboot启动netty的方式有很多：一种是实现 `CommandLineRunner` 接口；一种是配置bean，指定初始化方法；还有其他的

源码在GitHub上面

参考文章：crossoverjie.top/2018/05/24/...

> **关注下面的标签，发现更多相似文章**

Netty MyBatis Spring 数据库

Spring 中异常处理的各种姿势

9

专栏 · 中间件兴趣圈 · 1天前 · MyBatis / 后端

Mybatis执行SQL的4大基础组件详解(图文并茂)

3

专栏 · yanglbme · 5天前 · 数据库

面试官：分库分表之后，id 主键如何处理?

28

专栏 · Noodles_Mars · 4天前 · Spring / Java

手写Spring框架，加深对Spring工作机制的理解！

17

专栏 · Billions · 1天前 · 数据库

理解Google Spanner(3)：分布式事务原理与实现

2

专栏 · 胖先森 · 4天前 · Spring

SpringMVC注解版：读取核心配置类

2

专栏 · BUG9 · 3天前 · Spring

Mybatis源码解析(三) —— Mapper代理类的生成

3

专栏 · Van_ · 12天前 · Spring

你还在用BeanUtils进行对象属性拷贝?

23 5