

1、什么是Spring框架集成MyBatis框架？

意思是把Spring框架和MyBatis框架结合起来当做同一个框架使用，主要是用Spring集成mybatis，我们就只需要使用Spring就行了，不需要在独立使用mybatis。

实现方式：就是让mybatis中所必需的类给Spring容器创建、赋值、管理

mybatis中需要交给Spring容器创建与管理的类：

- 1、连接池对象，让Spring创建阿里的druid连接池，不使用mybatis默认的连接池对象
- 2、SqlSessionFactory对象
- 3、dao对象

需要学习的就是上面三个对象的创建语法，暂时使用配置文件的bean标签

2、Spring集成mybatis具体项目步骤

在idea-maven-spring工程中新建项目e-spring-mybatis-05

- 1、创建maven项目
- 2、引入依赖
 - 1)、spring依赖
 - 2)、mybatis依赖
 - 3)、mysql驱动依赖
 - 4)、spring的事务依赖
 - 5)、mybatis和spring集成的依赖：该依赖是mybatis官方提供的，用来在Spring项目中创建mybatis的SqlSessionFactory、dao对象
 - 6)、阿里的druid连接池依赖
- 3、创建实体类
- 4、创建dao接口和mapper文件
- 5、创建mybatis核心配置文件
- 6、创建Service接口和其实现类，属性是dao
- 7、创建Spring配置文件：在其中声明mybatis的一些对象由Spring创建
 - 1)、数据源，即DataSource对象
 - 2)、SqlSessionFactory
 - 3)、dao对象
 - 4) 声明的自定义的Service对象
- 8、创建测试类，获取Service对象，通过service调用dao完成数据库的访问

1)、创建resources资源目录。在pom文件中添加依赖，还需要配置Resource文件扫描插件，具体如下

```
<?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>com.studymyself</groupId>
    <artifactId>e-spring-mybatis-05</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
    </properties>

    <dependencies>
        <!--      单元测试依赖-->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.11</version>
            <scope>test</scope>
        </dependency>

        <!--      spring依赖, 核心IOC-->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.2.5.RELEASE</version>
        </dependency>
        <!--      下面两个做Spring事务的jar包, 因为要访问数据库, 事务相关-->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-tx</artifactId>
            <version>5.2.5.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-jdbc</artifactId>
            <version>5.2.5.RELEASE</version>
        </dependency>

        <!--      mybatis依赖-->
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis</artifactId>
            <version>3.5.3</version>
        </dependency>

        <!--      mybatis和Spring集成的依赖, 由mybatis官方提供-->
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis-spring</artifactId>
            <version>2.0.3</version>
        </dependency>
    </dependencies>
</project>
```

```

<!--    mysql驱动依赖-->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.13</version>
</dependency>
</dependencies>

<build>
<!--    因为mybatis的dao层存放映射文件，不放在resources目录下，所以要加resource插件指定扫描-->
    <resources>
        <resource>
            <!--表示编译时在src/main/java目录下的-->
            <directory>src/main/java</directory>
            <!--所有properties、xml类型的文件可以被扫描到,然后拷贝到编译生成文件夹的对应目录中-->
            <includes>
                <include>**/*.properties</include>
                <include>**/*.xml</include>
            </includes>
            <!--filtering的值false表示不启用过滤器，上面include已经是过滤操作了-->
            <filtering>false</filtering>
        </resource>
    </resources>
</build>
</project>

```

2) 、创建实体类com.studymyself.entity.User和dao接口com.studymyself.dao.UserDao以及mapper文件UserDao.xml

```

package com.studymyself.entity;

public class User {
    //是属性名和数据库表的列名一样
    private Integer id;
    private String loginName;
    private String loginPwd;
    private String realName;

    //为了方便赋值，可以把无参数构造方法和有参数构造方法创建出来
    public User() {
    }

    public User(Integer id, String loginName, String loginPwd, String realName)
    {
        this.id = id;
        this.loginName = loginName;
        this.loginPwd = loginPwd;
        this.realName = realName;
    }

    public Integer getId() {
        return id;
    }

```

```

    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getLoginName() {
        return loginName;
    }

    public void setLoginName(String loginName) {
        this.loginName = loginName;
    }

    public String getLoginPwd() {
        return loginPwd;
    }

    public void setLoginPwd(String loginPwd) {
        this.loginPwd = loginPwd;
    }

    public String getRealName() {
        return realName;
    }

    public void setRealName(String realName) {
        this.realName = realName;
    }

    @Override
    public String toString() {
        return "User{" +
            "id=" + id +
            ", loginName='" + loginName + '\'' +
            ", loginPwd='" + loginPwd + '\'' +
            ", realName='" + realName + '\'' +
            '}';
    }
}

```

```

package com.studymyself.dao;

import com.studymyself.entity.User;

import java.util.List;

public interface UserDao {
    //定义两个方法就行了

    //添加数据方法
    public Integer insertUser(User user);

    //查询方法
    public List<User> selectAll();
}

```

```

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
    PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
    "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.studymyself.dao.UserDao">

    <insert id="insertUser">
        insert into t_user(loginName,loginPwd,realName) values ({loginName},{
loginPwd},{realName})
    </insert>

    <select id="selectAll" resultType="User">
        select * from t_user order by loginName desc
    </select>
</mapper>

```

3) 、创建mybatis核心配置文件mybatis.xml

因为选择使用阿里的druid数据库连接池，所以原先配置在mybatis核心配置文件的DataSource标签就不需要了。

我们可以添加一个Spring集成mybatis专门使用的核心配置文件模板，File->Settings-File and Code Templates-Files,下面的+号->Name填: spring-mybatis-config , Extension填: xml ->模板内容如下:

```

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
    PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
    "http://mybatis.org/dtd/mybatis-3-config.dtd">

<configuration>

    <!--注意这些属性设置从上往下是有顺序的，写反就会报错-->

    <!--配置属性资源文件-->
    <properties resource="jdbc.properties"/>

    <!--该属性的作用：控制mybatis全局行为，
        顺序要在上面的properties属性下面-->
    <settings>
        <!--设置mybatis输出日志-->
        <setting name="logImpl" value="STDOUT_LOGGING"/>
    </settings>

    <!--给这个包下的所有类起别名-->
    <typeAliases>
        <package name="实体类包名"/>
    </typeAliases>

```

```

<!--配置插件-->
<plugins>
    <!--分页插件-->
    <plugin interceptor="com.github.pagehelper.PageInterceptor"></plugin>
</plugins>

<mappers>
    <!--第一种方式-->
    <!-- <mapper resource="mapper文件的类根路径"/>-->

    <!--第二种方式：使用包名
    其中name属性的值：mapper文件所在的包名
    使用package的条件：
        1、mapper文件的名称要和接口的名称一致，包括大小写一致
        2、mapper文件要和接口在同一个目录之中-->
    <package name="mapper文件所在接口的包名"/>
</mappers>
</configuration>

```

创建的mybatis核心配置文件如下：

```

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
    PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
    "http://mybatis.org/dtd/mybatis-3-config.dtd">

<configuration>

    <!--注意这些属性设置从上往下是有顺序的，写反就会报错-->

    <!--配置属性资源文件-->
    <properties resource="jdbc.properties"/>

    <!--该属性的作用：控制mybatis全局行为，
    顺序要在上面的properties属性下面-->
    <settings>
        <!--设置mybatis输出日志-->
        <setting name="logImpl" value="STDOUT_LOGGING"/>
    </settings>

    <!--给这个包下的所有类起别名-->
    <typeAliases>
        <package name="com.studymyself.entity"/>
    </typeAliases>

    <!--配置插件-->
    <plugins>
        <!--分页插件-->
        <plugin interceptor="com.github.pagehelper.PageInterceptor"></plugin>
    </plugins>

    <mappers>
        <!--第一种方式-->
        <!-- <mapper resource="mapper文件的类根路径"/>-->
    </mappers>

```

```
<!--第二种方式：使用包名
    其中name属性的值：mapper文件所在的包名
    使用package的条件：
        1、mapper文件的名称要和接口的名称一致，包括大小写一致
        2、mapper文件要和接口在同一个目录之中-->
    <package name="com.studymyself.dao"/>
</mappers>
</configuration>
```

4)、创建service类。新建一个service包，其中新建一个UserService接口和在子包impl中实现该接口UserServiceImpl

```
package com.studymyself.service;

import com.studymyself.entity.User;

import java.util.List;

public interface UserService {

    public Integer addUser(User user);

    public List<User> queryAll();

}
```

```
package com.studymyself.service;

import com.studymyself.dao.UserDao;
import com.studymyself.entity.User;

import java.util.List;

public class UserServiceImpl implements UserService {

    //定义dao接口类型属性
    UserDao userDao;

    //因为本项目不用注解注入，所以定义一个set方法
    public void setUserDao(UserDao userDao) {
        this.userDao = userDao;
    }

    @Override
    public Integer addUser(User user) {
        //在现实的开发中，service的方法中通常是做很多的判断以及很多
        //业务处理等功能的，然后才决定要不要插入数据的。在一个service
        //方法中可能会调用多个dao的方法，这里只是演示，所以就只有调用
        //dao接口中的方法
        int count = userDao.insertUser(user);
        return count;
    }

}
```

```

@Override
public List<User> queryAll() {
    List<User> userList = userDao.selectAll();
    return userList;
}
}

```

5)、准备创建Spring配置文件，由于不使用注解，所以所有有关对象的创建都需要在该配置文件中声明。但是我们使用的连接池对象是阿里的druid连接池，到底该连接池的具体实现类是怎样的我们要学习一下。druid项目在github上面，我们通过链接<https://github.com/alibaba/druid>可以到druid项目的主页，其中有很多关于druid的使用介绍，<https://github.com/alibaba/druid/wiki>链接可以查看配置方式如何编写，我们就进入这个链接来查看。

English | 中文

Druid 是一个 JDBC 组件库，包含数据库连接池、SQL Parser 等组件，被大量业务和技术产品使用或集成，经历过最严苛线上业务场景考验，是你值得信赖的技术产品。

下载

你可以从如下地方下载：

maven central repository <http://repo1.maven.org/maven2/com/alibaba/druid/>

Maven 依赖

```
<dependency>
  <groupId>com.alibaba</groupId>
  <artifactId>druid</artifactId>
  <version>1.1.12</version>
</dependency>
```

Gradle

```
compile group: 'com.alibaba', name: 'druid', version: '1.1.12'
```

Pages 77

Find a Page...

- Home
- Benchmark_aliyun
- Contributors
- DBCP迁移
- Druid Change Log
- druid monitor设计
- Druid_SQL_AST
- DruidDataSource
- DruidDataSource动态配置
- DruidDataSource统计数据日志输出格式
- DruidDataSource配置
- DruidDataSource配置属性列表
- Druid中使用log4j2进行日志输出
- Druid连接池介绍
- Druid锁的公平模式问题

Show 62 more pages...

主要查看这个

2020最新SSM框架全视频教程

DruidDataSource配置 - alibaba

github.com/alibaba/druid/wiki/DruidDataSource配置

应用 百度 淘宝 京东 天猫 苏宁易购 杂七杂八 tensorflow 工具 GitHub java学习 数据结构 计算机基础 算法 后端 python学习 tensorflow2.0的坑 其他书籍

DruidDataSource配置

温绍钢 edited this page on 14 Oct 2019 · 21 revisions

1. 通用配置

DruidDataSource大部分属性都是参考DBCP的，如果你原来就是使用DBCP，迁移是十分方便的。

```
<bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource" init-method="init" destroy-method="close">
  <property name="url" value="${jdbc_url}" />
  <property name="username" value="${jdbc_user}" />
  <property name="password" value="${jdbc_password}" />

  <property name="filters" value="stat" />

  <property name="maxActive" value="20" />
  <property name="initialSize" value="1" />
  <property name="maxWait" value="60000" />
  <property name="minIdle" value="1" />

  <property name="timeBetweenEvictionRunsMillis" value="60000" />
  <property name="minEvictableIdleTimeMillis" value="300000" />

  <property name="testWhileIdle" value="true" />
  <property name="testOnBorrow" value="false" />
  <property name="testOnReturn" value="false" />

  <property name="poolPreparedStatements" value="true" />
  <property name="maxOpenPreparedStatements" value="20" />

  <property name="asyncInit" value="true" />
</bean>
```

- 在上面的配置中，通常你需要配置url、username、password，maxActive这三项。
- Druid会自动跟url识别驱动类名，如果连接的数据库非常见数据库，配置属性driverClassName
- asyncInit是1.1.4中新增加的配置，如果有initialSize数量较多时，打开会加快应用启动时间

通用配置中我们可以看到，我们要想要让Spring容器创建该连接池对象，是非定义的类，需要按上面图中的编写配置。知道该连接池对象的类型是com.alibaba.druid.pool.DruidDataSource，mybatis中的是DataSource类。DruidDataSource类中有很多属性，需要我们赋值，具体如下：

```
<bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource"
      init-method="init" destroy-method="close">
  <!--
    说明：
      init-method="init": init是在DruidDataSource类中定义的一个初始方法，表示创建连接
      池对象之后执行该方法，该类的构造方法之后自动由Spring调用的
      destroy-method="close": close是在DruidDataSource类中定义的一个关闭资源的方法，
      当Spring容器关闭时自动调用该方法，进行资源的关闭
    -->

  <!--
    下面三个属性存储的就是连接数据库的三要素，从上面的说明中，我们不需要添加注册数据库的驱
    动的字符串：即之前在mybatis核心配置文件的这一项：<property name="driver"
    value="${jdbc.driver}"/>
    Druid会根据下面的url属性的值识别的，除非是不常见的，需要添加驱动属性
    driverClassName，然后注入
    从${jdbc_url}可以看出，通常这里的值可以写到属性配置文件中，更换不同数据库时方便修改。
    -->
    <property name="url" value="${jdbc_url}" />
    <property name="username" value="${jdbc_user}" />
    <property name="password" value="${jdbc_password}" />

    <!-- 下面的这些属性是连接池更加细微的控制条件 -->
    <property name="filters" value="stat" />

    <property name="maxActive" value="20" /> <!-- 表示连接池的最大容纳的连接对象
    Connection数量 -->
    <property name="initialSize" value="1" /> <!-- 表示连接池创建时的初始连接对象数量 -->
    <property name="maxwait" value="60000" /> <!-- 等待时间，单位毫秒 -->
    <property name="minIdle" value="1" />

    <property name="timeBetweenEvictionRunsMillis" value="60000" />
    <property name="minEvictableIdleTimeMillis" value="300000" />

    <property name="testWhileIdle" value="true" />
    <property name="testOnBorrow" value="false" />
    <property name="testOnReturn" value="false" />

    <property name="poolPreparedStatements" value="true" />
    <property name="maxOpenPreparedStatements" value="20" />

    <property name="asyncInit" value="true" />
</bean>
```

最后，Spring的配置文件applicationContext.xml如下所示

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

    <!--声明数据源DataSource对象，作用是连接数据库-->
    <bean id="myDataSource" class="com.alibaba.druid.pool.DruidDataSource"
        init-method="init" destroy-method="close">
<!--      使用set注入赋值给DruidDataSource提供连接数据库否信息-->
        <property name="url" value="jdbc:mysql://localhost:3306/mysqlstudy?
useSSH=false"/>
        <property name="username" value="root"/>
        <property name="password" value="rong195302"/>

        <property name="maxActive" value="20"/>
    </bean>

<!--      声明mybatis提供的SqlSessionFactoryBean类对象，这个类内部有一个方法是创建
SqlSessionFactory-->
<!--      对象的，这样我们声明SqlSessionFactoryBean类对象就是声明SqlSessionFactory对象
了。就像之前-->
<!--      使用的SqlSessionFactoryBuilder类中的build方法一样-->
    <bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
        <!--我们知道之前创建SqlSessionFactory对象时需要使用到mybatis的核心配置文件，而
        配置文件中的核心配置中没有了。改为上面的声明对象了，所以SqlSessionFactoryBean
类中
        有一个属性存储数据库信息，这时我们就把上面的数据源对象放到该属性中-->

        <!--通过set注入把数据库连接池赋给了dataSource属性,引用类型属性注意用ref-->
        <property name="dataSource" ref="myDataSource"/>

        <!--下面就是把核心配置文件读取到这个类中的configLocation属性中，属性是Resource类
型
        的，就像之前用mybatis中获取核心配置文件时Resources.getResourceAsStream一
样，是
        读取配置文件的,注意注入值是文件时要在前面添加路径标识：classpath-->
        <property name="configLocation" value="classpath:mybatis.xml"/>

        <!--最后创建的SqlSessionFactory对象类名是DefaultSqlSessionFactory,这是
SqlSessionFactory接口的实现类-->
    </bean>

<!--      这里就是创建dao对象，或者dao的代理对象，使用SqlSession的getMapper(接口的Class对
象)
        我们不能为每一个dao接口去一个一个调用getMapper方法生成代理对象，所以声明某个类对象
        一次性把所有的dao接口都生成代理对象，如下
        MapperScannerConfigurer这个类：可以一次性把符合条件的dao接口都生成其代理对象
        其中每个代理对象存储在Spring容器集合中的key是接口的首字母小写-->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <!--之前我们要调用getMapper方法需要用到sqlSessionFactory对象和dao接口的Class对
        象
        所以MapperScannerConfigurer中有两个属性要获取这些数据-->

        <!--sqlSessionFactory对象使用上面的声明的bean，用set注入-,属性是String类型，用
        value-->
        <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory"/>

```

```

        <!--这里直接把dao接口的包名赋值给获取dao接口Class对象的属性，一次性创建所有dao接口
        的代理对象
        MapperScannerConfigurer会扫描这个包装的所有接口，为每个接口都执行一次
        getMapper方法，得到
        的每个接口的代理对象都放到Spring容器中
        -->
        <property name="basePackage" value="com.studymyself.dao"/>
    </bean>

    <!-- 声明service类对象-->
    <bean id="userService" class="com.studymyself.service.UserServiceImpl">
        <property name="userDao" ref="userDao"/>
    </bean>

</beans>

```

创建一个测试类，获取Spring容器中对象的名称和对象内存地址输出

```

package com.studymyself;

import org.junit.Test;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MyTest01 {

    @Test
    public void test(){

        //定义配置文件路径
        String config = "applicationContext.xml";

        //创建Spring容器对象
        ApplicationContext ac = new ClassPathXmlApplicationContext(config);

        //获取容器中的所有对象
        String[] beanNames = ac.getBeanDefinitionNames();

        //遍历beanNames
        for (String beanName:
            beanNames) {
            System.out.println("容器中的对象名称: "+beanName+" =
            "+ac.getBean(beanName));
        }

    }

}

```

结果如下


```
}  
  
}
```

7)、可以添加属性配置文件，其中存储连接数据库的信息

jdbc.properties

```
jdbc.url=jdbc:mysql://localhost:3306/mysqlstudy?useSSH=false  
jdbc.username=root  
jdbc.password=rong195302
```

因为连接数据库的信息不在在mybatis核心配置文件中，所以该配置文件不需要添加属性资源的配置。我们要在Spring配置文件中声明，让Spring知道属性资源配置文件的位置

```
<!--让spring知道jdbc.properties文件位置-->  
<context:property-placeholder location="classpath:jdbc.properties"/>
```