Day05

Spring 整合 MyBatis

http://mybatis.org/spring/index.html

https://docs.spring.io/spring-framework/docs/current/reference/html/data-access.html#jdbc

创建 Maven 项目

添加依赖

XML 复制代码

```xml
<?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.rushuni</groupId>
    <artifactId>spring-mybatis-demo-01</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>11</maven.compiler.source>
        <maven.compiler.target>11</maven.compiler.target>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.3.9</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-beans</artifactId>
            <version>5.3.9</version>
        </dependency>
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis</artifactId>
            <version>3.5.7</version>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.25</version>
        </dependency>
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>druid</artifactId>
            <version>1.2.6</version>
        </dependency>
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis-spring</artifactId>
            <version>2.0.6</version>
        </dependency>
        <dependency>
```

```
            <groupId>org.springframework</groupId>
            <artifactId>spring-jdbc</artifactId>
            <version>5.3.9</version>
        </dependency>
    </dependencies>

</project>
```

数据表

SQL 复制代码

```sql
create table account (
  id int primary key auto_increment,
  name varchar(100),
  money double
);
```

实体类

Java 复制代码

```java
public class Account implements Serializable {
    private Integer id;
    private String name;
    private Double money;
    // ...
}
```

数据层

Java 复制代码

```java
package com.rushuni.mapper;

import com.rushuni.entity.Account;

import java.util.List;

/**
 * @author rushuni
 * @date 2021年07月16日 10:51 上午
 */
public interface AccountMapper {

    void save(Account account);

    void delete(Integer id);

    void update(Account account);

    List<Account> listAll();

    Account getById(Integer id);
}
```

业务层

接口

Java 复制代码

```java
package com.rushuni.service;

import com.rushuni.entity.Account;

import java.util.List;

/**
 * @author rushuni
 * @date 2021年07月16日 10:52 上午
 */
public interface AccountService {

    void save(Account account);

    void delete(Integer id);

    void update(Account account);

    List<Account> listAll();

    Account getById(Integer id);
}
```

实现

Java 复制代码

```java
package com.rushuni.service.impl;

import com.rushuni.entity.Account;
import com.rushuni.mapper.AccountMapper;
import com.rushuni.service.AccountService;

import java.util.List;

/**
 * @author rushuni
 * @date 2021年07月16日 10:53 上午
 */
public class AccountServiceImpl implements AccountService {

    AccountMapper accountMapper;

    public AccountMapper getAccountMapper() {
        return accountMapper;
    }

    public void setAccountMapper(AccountMapper accountMapper) {
        this.accountMapper = accountMapper;
    }

    @Override
    public void save(Account account) {

    }

    @Override
    public void delete(Integer id) {

    }

    @Override
    public void update(Account account) {

    }

    @Override
    public List<Account> listAll() {
        return null;
    }

    @Override
    public Account getById(Integer id) {
        return null;
    }
```

```
49  }
```

jdbc.propertis

XML | 复制代码

```
1  jdbc.driver=com.mysql.cj.jdbc.Driver
2  jdbc.url=jdbc:mysql://localhost:3306/mybatis_demo
3  jdbc.username=xxx
4  jdbc.password=xxx
```

applicationContext

XML 复制代码

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

    <!-- 1.加载 perperties 配置文件 -->
    <context:property-placeholder location="classpath:*.properties"/>

    <!-- 2.配置 druid 数据库连接池 -->
    <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">
        <property name="driverClassName" value="${jdbc.driver}"/>
        <property name="url" value="${jdbc.url}"/>
        <property name="username" value="${jdbc.username}"/>
        <property name="password" value="${jdbc.password}"/>
    </bean>

    <!-- 3.配置 Service 层的 bean，注入 bean 依赖的 Mapper 对象 -->
    <bean id="accountService"
class="com.rushuni.service.impl.AccountServiceImpl">
        <property name="accountMapper" ref="accountMapper"/>
    </bean>

    <!-- 4.spring  整合 mybatis 配置 -->
    <bean class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource"/>
        <property name="typeAliasesPackage" value="com.rushuni.entity"/>
        <property name="mapperLocations" value="mapper/*.xml"/>
    </bean>

    <!-- 5.扫描 mapper 层，把 mapper 层的对象交给 Spring 管理，在第 3 步直接使用。-->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="basePackage" value="com.rushuni.mapper"/>
    </bean>

</beans>
```

测试

Java 复制代码

```java
package com.rushuni;

import com.rushuni.entity.Account;
import com.rushuni.service.AccountService;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

/**
 * @author rushuni
 * @date 2021年07月16日 10:47 上午
 */
public class Application {
    public static void main(String[] args) {
        ApplicationContext ctx = new ClassPathXmlApplicationContext("applicationContext.xml");
        AccountService accountService = (AccountService) ctx.getBean("accountService");
        Account account = new Account();
        account.setId(null);
        account.setMoney(50.0);
        account.setName("李四");
        accountService.save(account);
        account = accountService.getById(1);
        System.out.println(account);
    }
}
```

总结

- Spring 整合 MyBatis，需要使用 MyBatis 提供的 MyBatis-Spring 这个 jar 包。
- 整合后，因为资源都交给 Spring 管理，所以 Mybatis 核心配置文件也不需要了。
  - mybatis-config.xml 文件中的 environment 标签，在 Spring 中使用一个 bean 替代了。
  - 读取 mapper 映射文件的工作可以交给了 Spring 处理。
  - 类型别名的设置可以交由 Spring 处理。

课堂作业

- 完成 Spring 整合 MyBatis 案例。
- 完成增删改查操作。

Spring 注解开发

启动注解扫描（XML）

在 Spring 项目中使用注解通常会启动注解扫描功能：

```xml
<context:component-scan base-package="packageName"/>
```

定义 bean 的注解（@Controller | @Service | @Repository | @Component）

要把我们的类交给 Spring 容器去处理有以下几个注解：

- @Component
- @Controller
- @Service
- @Repository

把这些注解放在类的上方就可以了。

@Controller，@Service ，@Repository 是来自 @Component 的，功能相同其实可以互换。

这些注解可以使用 value 来定义 bean 的 id。

作用域注解（@scope）

作用域注解是 @Scope，放在类的上方用于设置 bean 的作用域，和 xml 中配置作用域一样，默认 value 是 sigleton 的。

生命周期注解（@PostConstruct、@PreDestroy）

和 xml 配置 bean 一样，可以使用设置生命周期中的初始化和销毁时只需的方法。

注解是 @PostConstruct、@PreDestroy。

列子：

```java
@PostConstruct
public void init() {
    System.out.println("init...");
}
@PreDestory
public void cleanup() {
  System.out.println("destory...");
}
```

读取第三方资源

通过 @bean 可以引入第三方的 bean。

注意：@Bean所在的类必须被spring扫描加载，否则该注解无法生效。

它 的 value默认值是定义 bean 的访问 id。

非引用类型注入

通过在属性和方法定义的上面添加 @Value 注解可以设置对应属性的值或对方法进行传参。

说明：

- value值仅支持非引用类型数据，赋值时对方法的所有参数全部赋值
- value值支持读取properties文件中的属性值，通过类属性将properties中数据传入类中
- value值支持SpEL
- @value注解如果添加在属性上方，可以省略set方法（set方法的目的是为属性赋值）

引用类型注入

如果是引用类型，则使用 @Autowired、@Qualifier。

和非引用类型注入一样，可以设置对应属性的对象或对方法进行引用类型传参。

```java
@Autowired(required = false)
@Qualifier("userDao")
private UserDao userDao;
```

@Autowired 是默认按类型装配。

使用@Qualifier，则可以指定自动装配的 bean 的 id。

属性 required 用于定义该属性是否允许为 null。

引用类型注入（非 Spring 规范）

引用类型注入还可以使用 @Inject、@Named、@Resource。

其中，@Inject 与 @Named 是 JSR330 规范中的注解，功能与 @Autowired 和 @Qualifier 完全相同，它们适用于不同架构场景。

@Resource是JSR250规范中的注解，它主要可以简化书写格式，比如可以通过 name 属性设置设置 bean 的 id，tpye 属性直接设置 bean 的类型。

加载 properties 注解

使用 @PropertySource 属性值可以加载properties文件中的属性值

Java 复制代码

```java
@PropertySource(value = "classpath:filename.properties")
public class ClassName {
    @Value("${propertiesAttributeName}")
    private String attributeName;
}
```

其中 value 的值默认为要加载的 properties 文件名。

还可以设置 ignoreResourceNotFound，如果资源未找到，是否忽略，默认为false。

注意：不支持*通配格式，一旦加载，所有spring控制的bean中均可使用对应属性值。

在类中启动注解扫描

在类中使用 @Configuration、@ComponentScan 注解可以设置当前类为 spring 核心配置加载类。

Java 复制代码

```java
@Configuration
@ComponentScan("scanPackageName")
public class SpringConfigClassName{
}
```

核心配合类用于替换spring核心配置文件，此类可以设置空的，不设置变量与属性。

bean扫描工作使用注解@ComponentScan替代。

此时获取 Spring context 需要使用 AnnotationConfigApplicationContext 类。

```java
AnnotationConfigApplicationContext ctx =
    new AnnotationConfigApplicationContext(SpringConfig.class);
```

bean 导入注解

@Import 注解可以导入第三方 bean 作为 spring 控制的资源。

```java
@Configuration
@Import(OtherClassName.class)
public class ClassName {
}
```

@Import 注解在同一个类上，仅允许添加一次，如果需要导入多个，使用数组的形式进行设定。

在被导入的类中可以继续使用 @Import 导入其他资源。

@Bean所在的类可以使用导入的形式进入spring容器，无需声明为 bean。

bean 加载控制

@DependsOn 注解可以控制 bean 的加载顺序，使其在指定 bean 加载完毕后再加载。

```java
@DependsOn("beanId")
public class ClassName {
}
```

@DependsOn的 bean 优先级高于 @Bean 和 @Component 配置的bean。

@Lazy 控制bean的加载时机，使其延迟加载。

```java
@Lazy
public class ClassName {
}
```

@Order 可以控制配置类的加载顺序。

```java
@Order(1)
public class SpringConfigClassName {
}
```

应用场景

@DependsOn的例子：

- 微信订阅号，发布消息和订阅消息的bean的加载顺序控制
- 双11活动期间，零点前是结算策略A，零点后是结算策略B，策略B操作的数据为促销数据。策略B加载顺序与促销数据的加载顺序

@Lazy

- 程序灾难出现后对应的应急预案处理是启动容器时加载时机

@Order

- 多个种类的配置出现后，优先加载系统级的，然后加载业务级的，避免细粒度的加载控制

课后作业

- 创建一个 maven 项目，整合 Spring 和 MyBatis，实现CRUD。
  - Spring 使用注解实现。
  - 数据表可以用之前的 Account 表。