参考文档
简介
入门
SqlSessionFactoryBean
事务
标准配置
交由容器管理事务
编程式事务管理
使用 SqlSession
注入映射器
使用 Spring Boot
使用 MyBatis API
使用 Spring Batch
示例代码
项目文档
项目信息
项目报表

事务

一个使用 MyBatis-Spring 的其中一个主要原因是它允许 MyBatis 参与到 Spring 的事务管理中。而不是给 MyBatis 创建一个新的专用事务管理器，MyBatis-Spring 借助了 Spring 中的 `DataSourceTransactionManager` 来实现事务管理。

一旦配置好了 Spring 的事务管理器，你就可以在 Spring 中按你平时的方式来配置事务。并且支持 `@Transactional` 注解和 AOP 风格的配置。在事务处理期间，一个单独的 `SqlSession` 对象将会被创建和使用。当事务完成时，这个 session 会以合适的方式提交或回滚。

事务配置好了以后，MyBatis-Spring 将会透明地管理事务。这样在你的 DAO 类中就不需要额外的代码了。

标准配置

要开启 Spring 的事务处理功能，在 Spring 的配置文件中创建一个 `DataSourceTransactionManager` 对象：

```
<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
  <constructor-arg ref="dataSource" />
</bean>
```

```
@Configuration
public class DataSourceConfig {
  @Bean
  public DataSourceTransactionManager transactionManager() {
    return new DataSourceTransactionManager(dataSource());
  }
}
```

传入的 `DataSource` 可以是任何能够与 Spring 兼容的 JDBC `DataSource`。包括连接池和通过 JNDI 查找获得的 `DataSource`。

注意：为事务管理器指定的 `DataSource` **必须**和用来创建 `SqlSessionFactoryBean` 的是同一个数据源，否则事务管理器就无法工作了。

交由容器管理事务

如果你正使用一个 JEE 容器而且想让 Spring 参与到容器管理事务（Container managed transactions，CMT）的过程中，那么 Spring 应该被设置为使用 `JtaTransactionManager` 或由容器指定的一个子类作为事务管理器。最简单的方式是使用 Spring 的事务命名空间或使用 `JtaTransactionManagerFactoryBean`：

```
<tx:jta-transaction-manager />
```

```
@Configuration
public class DataSourceConfig {
  @Bean
  public JtaTransactionManager transactionManager() {
    return new JtaTransactionManagerFactoryBean().getObject();
  }
}
```

在这个配置中，MyBatis 将会和其它由容器管理事务配置的 Spring 事务资源一样。Spring 会自动使用任何一个存在的容器事务管理器，并注入一个 `SqlSession`。如果没有正在进行的事务，而基于事务配置需要一个新的事务的时候，Spring 会开启一个新的由容器管理的事务。

注意，如果你想使用由容器管理的事务，而不想使用 Spring 的事务管理，你就**不能**配置任何的 Spring 事务管理器。并**必须**配置 `SqlSessionFactoryBean` 以使用基本的 MyBatis 的 `ManagedTransactionFactory`：

```
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
  <property name="dataSource" ref="dataSource" />
  <property name="transactionFactory">
    <bean class="org.apache.ibatis.transaction.managed.ManagedTransactionFactory" />
  </property>
</bean>
```

```
@Configuration
public class MyBatisConfig {
  @Bean
  public SqlSessionFactory sqlSessionFactory() {
    SqlSessionFactoryBean factoryBean = new SqlSessionFactoryBean();
    factoryBean.setDataSource(dataSource());
    factoryBean.setTransactionFactory(new ManagedTransactionFactory());
    return factoryBean.getObject();
  }
}
```

编程式事务管理

MyBatis 的 `SqlSession` 提供几个方法来在代码中处理事务。但是当使用 MyBatis-Spring 时，你的 bean 将会注入由 Spring 管理的 `SqlSession` 或映射器。也就是说，Spring 总是为你处理了事务。

你不能在 Spring 管理的 `SqlSession` 上调用 `SqlSession.commit()`，`SqlSession.rollback()` 或 `SqlSession.close()` 方法。如果这样做了，就会抛出 `UnsupportedOperationException` 异常。在使用注入的映射器时，这些方法也不会暴露出来。

无论 JDBC 连接是否设置为自动提交，调用 `SqlSession` 数据方法或在 Spring 事务之外调用任何在映射器中方法，事务都将会自动被提交。

如果你想编程式地控制事务，请参考 the Spring reference document(Data Access -Programmatic transaction management-)。下面的代码展示了如何使用 `PlatformTransactionManager` 手工管理事务。

```
public class UserService {
  private final PlatformTransactionManager transactionManager;
  public UserService(PlatformTransactionManager transactionManager) {
    this.transactionManager = transactionManager;
  }
  public void createUser() {
    TransactionStatus txStatus =
        transactionManager.getTransaction(new DefaultTransactionDefinition());
    try {
      userMapper.insertUser(user);
    } catch (Exception e) {
      transactionManager.rollback(txStatus);
      throw e;
    }
    transactionManager.commit(txStatus);
  }
}
```

在使用 `TransactionTemplate` 的时候，可以省略对 `commit` 和 `rollback` 方法的调用。

```
public class UserService {
  private final PlatformTransactionManager transactionManager;
  public UserService(PlatformTransactionManager transactionManager) {
    this.transactionManager = transactionManager;
  }
  public void createUser() {
    TransactionTemplate transactionTemplate = new TransactionTemplate(transactionManager);
    transactionTemplate.execute(txStatus -> {
      userMapper.insertUser(user);
      return null;
    });
  }
}
```

注意：虽然这段代码使用的是一个映射器，但换成 SqlSession 也是可以工作的。