企业级框架 Spring

Unit02

MyBatis简介

MyBatis简介

- MyBatis 最早源自Apache基金会的一个开源项目iBatis, 2010年这个项目由Apache software foundation 迁移到了google code，并且改名为MyBatis；
- MyBatis是支持普通SQL查询，存储过程和高级映射的ORM框架。
- MyBatis封装了几乎所有的JDBC代码和参数的手工设置以及结果集的检索；

MyBatis简介

- MyBatis框架主要结构
  - 实体类： 封装记录信息
  - SQL定义文件：定义sql语句
  - 主配置文件：定义连接信息、加载SQL文件
  - 框架API：用于实现数据库增删改查操作

MyBatis简介

* Emp实体类的示例代码

```
public class Emp implements Serializable{
private int id;
private String name;
private double salary;
public int getId() {
        return id;
}
public void setId(int id) {
        this.id = id;
}
//省略其他setter和getter方法
```

MyBatis简介

- SQL定义文件的示例代码（EmpMapper.xml）

```
<mapper>
    <insert id="addEmp“ parameterType=“com.xdl.entity.Emp">
        insert into EMP (ID, NAME,SALARY)
        values (#{id},#{name},#{salary})
    </insert>
    <select id="findAll" resultType=" com.xdl.entity.Emp ">
        select * from EMP
    </select>
</mapper>
```

MyBatis简介

- 主配置文件的示例代码(SqlMapConfig.xml)

```
<configuration>
<environments default="environment">
   <environment id="environment">
      <transactionManager type="JDBC" />
      <dataSource type="POOLED">
         <property name="driver"
            value="oracle.jdbc.OracleDriver"/>
         <property name="url"
            value="jdbc:oracle:thin:@localhost:tarena10g" />
         <property name="username" value="demo" />
         <property name="password" value="demo" />
      </dataSource>
   </environment>
</environments>
<mappers>
  <mapper resource="com/xdl/mapper/EmpMapper.xml" />
</mappers>
</configuration>
```

MyBatis简介

- 框架提供的主要API如下
  - SqlSessionFactoryBuilder

  负责加载SqlMapConfig.xml构建SqlSessionFactory
  - SqlSessionFactory

  负责创建SqlSession对象实例
  - SqlSession

  负责执行SQL操作，用于执行已映射的SQL语句

MyBatis应用

MyBatis应用

- MyBatis应用步骤如下：
  - 为工程添加MyBatis开发包和数据库驱动包；
  - 在src下添加主配置文件SqlMapConfig.xml，定义连接参数；
  - 根据要操作数据表定义实体类
  - 编写SQL定义文件，定义SQL语句（在SqlMapConfig.xml指定加载关系）
  - 利用框架API编程，获取SqlSession实例

MyBatis应用

- 获取SqlSession实例的示例代码

```
String conf = "SqlMapConfig.xml";
Reader reader = Resources.getResourceAsReader(conf);

//创建SessionFactory对象
SqlSessionFactoryBuilder sfb = new SqlSessionFactoryBuilder();
SqlSessionFactory sf = sfb.build(reader);

//创建Session
SqlSession session = sf.openSession();
```

MyBatis应用

- SqlSession对象的操作方法如下：
  - insert(..) 插入操作
  - update(..) 更新操作
  - delete(..) 删除操作
  - selectOne(..) 单行查询操作
  - selectList(..) 多行查询操作

MyBatis应用

* Mapper映射器是用于映射SQL语句的接口，SqlSession可以根据映射器接口自动生成实现类，并创建出对象。

```
SqlSession session = factory.openSession();
//利用mapper接口获取实例
EmpMapper mapper =session.getMapper(EmpMapper.class);
mapper.save(emp);
session.close();
```

MyBatis应用

- Mapper映射器接口的规则如下
  - 方法名称要和SqlMapper.xml中的SQL的id保持一致
  - 方法参数要和SqlMapper.xml中的SQL的parameterType保持一致
  - 方法返回值要参考SqlMapper.xml中的resultType
    - 增删改返回值可以是int或void
    - 单行查询返回类型与resultType一致
    - 多行查询返回类型为List<resultType类型>

MyBatis应用

- 在SQL定义中，resultType属性用于指定查询数据采用哪种类型封装，规则为结果集列名和属性名一致，如果不一致将不能接收查询结果。
- 上述问题的解决方法有两种
  - 使用别名，select语句使用与属性一致的别名
  - 使用resultMap指定结果集列名和属性名的对应关系

MyBatis应用

- 使用resultMap自定义封装规则，示例代码

```
<select id="findAll" resultMap="empMap">
    select ENO,ENAME,SALARY from EMP
</select>

<resultMap id="empMap" type="org.tarena.entity.Emp">
    <result property="id" column="ENO"/>
    <result property="name" column="ENAME"/>
    <result property="sal" column="SALARY"/>
</resultMap>
```

Spring+MyBatis应用

Spring+MyBatis应用

Spring与MyBatis整合需要引入一个mybatis-spring.jar文件包，该包提供了下面几个与整合相关的API

* SqlSessionFactoryBean

  创建SqlSessionFactory对象，为整合应用提供SqlSession对象资源

* MapperFactoryBean

  根据指定的某一个Mapper接口生成Bean实例

* MapperScannerConfigurer

  根据指定包批量扫描Mapper接口并生成实例

* SqlSessionTemplate

  类似于JdbcTemplate，便于程序员自己编写Mapper实现类

Spring+MyBatis应用

- SqlSessionFactoryBean使用示例

```
<bean id="sqlSessionFactory"
        class="org.mybatis.spring.SqlSessionFactoryBean">
    <!-- 指定连接资源 -->
    <property name="dataSource" ref="myDataSource" />
   <!-- 指定映射文件 -->
    <property name="mapperLocations"
                value="classpath:com/xdl/entity/*.xml"/>
</bean>
```

Spring+MyBatis应用

- MapperFactoryBean使用示例

```
<bean id=“empMapper"
class="org.mybatis.spring.mapper.MapperFactoryBean">
    <property name="mapperInterface"
        value=“com.xdl.mapper.EmpMapper" />
    <property name="sqlSessionFactory"
                ref="sqlSessionFactory" />
  </bean>
```

Spring+MyBatis应用

- MapperScannerConfigurer使用示例

```
<bean
class="org.mybatis.spring.mapper.MapperScannerConfigurer">

      <property name="basePackage" value="com.xdl.mapper" />

    <property name="annotationClass"
        value="com.xdl.annotation.MyBatisRepository"/>
</bean>
```

Spring+MyBatis应用

- SqlSessionTemplate使用示例-1

```
<bean id="sqlSessionTemplate"
        class="org.mybatis.spring.SqlSessionTemplate">
    <constructor-arg index="0" ref="sqlSessionFactory">
    </constructor-arg>
</bean>
```

Spring+MyBatis应用

- SqlSessionTemplate使用示例-2

```
@Repository
public class MyBatisEmpDAO implements EmpDAO{
        @Resource
        private SqlSessionTemplate template;

        public List<Emp> findAll() {
                List<Emp> list = template.selectList("findAll");
                return list;
        }
}
```

总结和答疑