### Hibernate

* 对象映射配置

<hibernate-mapping>

<class name="com.hbnu.pojo.Player" table="tb\_player">

<id name="pid">

<generator class="native"></generator>

</id>

<property name="name" />

<property name="gender" />

<!-- table属性值表示多对多中第三张中间表的名称 -->

<set name="roles" table="tb\_pr" cascade="save-update">

<!-- key标签中的属性column值表示当前配置文件对应的表在第三张中间表的外键名称 -->

<key column="pid"></key>

<!-- 多对多

class:表示与当前配置文件对应的实体类全路径名称

column:表示多对多中另外一方对应的表在第三张中间表的外键名称

-->

<many-to-many class="com.hbnu.pojo.Role" column="rid" />

</set>

</class>

</hibernate-mapping>

* 工具类

public class HibernateUtil {

private static Configuration configuration;

private static SessionFactory sessionFactory;

static {

configuration = new Configuration().configure();

sessionFactory = configuration.buildSessionFactory();

}

public static SessionFactory getSessionFactory() {

return sessionFactory;

}

public static Session getSession() {

return sessionFactory.getCurrentSession(); // 获取与本地线程绑定的session对象

}

public static void main(String[] args) {

}

}

* 事务代码 并测试工具类

public void transactionTest() {

SessionFactory sessionFactory = null;

Session session = null;

Transaction transaction = null;

try {

sessionFactory = HibernateUtil.getSessionFactory();

session = sessionFactory.openSession();

transaction = session.beginTransaction();

User user = session.get(User.class, 1);

user.setAddress("湖北黄石");

transaction.commit();

} catch (HibernateException e) {

e.printStackTrace();

if (transaction != null) {

transaction.rollback();

}

} finally {

if (session != null) {

session.close();

}

if (sessionFactory != null) {

sessionFactory.close();

}

}

}

### Mybatis

* Mybatis-config.xml

<?xml version="1.0" encoding="utf-8" ?>

<!DOCTYPE configuration

PUBLIC "-//mybatis.org//DTD Config 3.0//EN"

"https://mybatis.org/dtd/mybatis-3-config.dtd">

<!-- MyBatis全局配置文件 -->

<configuration>

<!-- 引入配置文件资源 -->

<properties resource="db.properties" />

<!-- 配置别名 -->

<typeAliases>

<!-- 给某个具体的类设置别名 -->

<!--<typeAlias type="com.hbnu.pojo.User" alias="\_User" />-->

<!-- 给某个包下所有的类进行别名设置，设置后的别名为实体类类名 -->

<package name="com.hbnu.pojo"/>

</typeAliases>

<!-- environments标签可以配置多个环境，比如开发环境、测试环境、生产环境，但是同一时刻只能使用一个环境，具体使用哪个环境，通过default属性决定

default属性值表明使用的是哪个具体的环境

-->

<environments default="develop">

<!-- 具体环境配置信息 id属性值表示当前环境的标识符 -->

<environment id="develop">

<!-- transactionManager表示事务管理配置，type属性值有以下俩个

JDBC:属性值为JDBC表示开启自动事务管理机制，通过JDBC数据源管理事务范围，推荐使用

MANAGER:该属性值表示手动开启事务管理

-->

<transactionManager type="JDBC"></transactionManager>

<!-- 数据源，type属性值有以下3个

JNDI:已过时，不使用

POOLED:表示使用JDBC数据库连接池

UNPOOLED:表示不使用数据库连接池

-->

<dataSource type="POOLED">

<property name="driver" value="${driver}"/>

<property name="url" value="${url}"/>

<property name="username" value="${username}"/>

<property name="password" value="${password}"/>

</dataSource>

</environment>

</environments>

<!-- mappers表示映射配置文件信息，可以引入多个映射配置文件 -->

<mappers>

<mapper resource="UserMapper.xml" />

<mapper resource="ClassesMapper.xml" />

<mapper class="com.hbnu.dao.UserMapper" />

</mappers>

</configuration>

* db.properties

driver=com.mysql.cj.jdbc.Driver

url=jdbc:mysql://localhost:3306/class2013?serverTimezone=GMT&useSSL=false

username=root

password=chendikai

public class MyBatisTest {

private SqlSession sqlSession = null;

@Before //单元测试的时候会先执行这个方法初始化

public void beforeCRUD() throws IOException {

// 1、获取配置文件输入流对象

InputStream inputStream = Resources.getResourceAsStream("mybatis-config.xml");

// InputStream inputStream = MyBatisTest.class.getClassLoader().getResourceAsStream("mybatis-config.xlml");

// 2、通过配置文件输入流对象构建会话工厂

SqlSessionFactory sqlSessionFactory = new SqlSessionFactoryBuilder().build(inputStream);

// 3、通过会话工厂创建会话对象

sqlSession = sqlSessionFactory.openSession();

}

public void findById2() { //调用Mapper 接口定义的sql

User user = sqlSession.selectOne("UserMapper.findById", 3);

System.out.println(user);

}

}

* Mapper 接口定义sql

public interface UserMapper {

@Select("select \* from tb\_user where id = #{id}")

public abstract User findById(int id);

}

* 动态sql foreach

<select id="findAll6" resultMap="userResult">

select \* from tb\_user where username in

<foreach collection="array" open="(" close=")" separator="," item="name">

#{name}

</foreach>

</select>

public void findAll6() {

String[] usernames = {"阿凯", "阿凯2"};

List<User> users = sqlSession.selectList("UserMapper.findAll6", usernames);

for (User user : users) {

System.out.println(user);

}

}

* 动态sql if

<update id="updateData5">

update tb\_user

<set>

<if test="name != null">

username = #{name},

</if>

<if test="addr != null">

address = #{addr},

</if>

<if test="salary != null">

salary = #{salary}

</if>

</set>

where id = #{id}

</update>

@Test

public void updateData5() {

User user = new User();

user.setId(3);

user.setName("凯die");

user.setAddr("湖北武汉");

user.setSalary(12000.00);

sqlSession.update("UserMapper.updateData5", user);

}

* 删改查

<insert id="addData2">

insert into tb\_user(name, addr) values(#{name}, #{addr})

</insert>

<update id="updateData2">

update tb\_user set name = #{name} where id = #{id}

</update>

<delete id="deleteData2">

delete from tb\_user where id = #{id}

</delete>

public void addData2() {

User user = new User();

user.setName("臣弟阿凯");

user.setAddr("湖北黄石");

sqlSession.insert("UserMapper.addData2", user);

}

public void updateData2() {

User user = new User();

user.setId(5);

user.setName("黄石阿凯");

sqlSession.update("UserMapper.updateData2", user);

}

public void deleteData2() {

sqlSession.delete("UserMapper.deleteData2", 5);

}

* resultMap 复杂对象

public class Classes {

private int cid;

private String cname;

Set<Teacher> teachers = new HashSet<>();

}

<resultMap id="classesResult" type="com.hbnu.pojo.Classes">

<id property="cid" column="cid" />

<result property="cname" column="cname" />

<association property="teacher" javaType="com.hbnu.pojo.Teacher">

<id property="tid" column="tid" />

<result property="tname" column="tname" />

</association>

</resultMap>

<!-- 3、根据老师id获取老师多带的班级信息(多对多) -->

<select id="getClassesByTeacherId2" resultMap="classesResult2">

select \* from tb\_classes c, tb\_teacher t, tb\_tc tc

where c.cid = tc.cid

and t.tid = tc.tid

and t.tid = #{t.tid}

</select>

### SSM注解配置

* 配置mybatis

@Configuration

@MapperScan("com.hbnu.dao") //扫描mybatis mapper

public class SpringRepositoryConfig {

@Bean(value = "dataSource", initMethod = "init", destroyMethod = "close")

public DruidDataSource dataSource() {

DruidDataSource dataSource = new DruidDataSource(); dataSource.setUrl("jdbc:mysql://localhost:3306/class2013?serverTimezone=GMT&useSSL=false");

dataSource.setUsername("root");

dataSource.setPassword("chendikai");

return dataSource;

}

@Bean("sqlSessionFactory")

public SqlSessionFactory newSqlSessionFactory(DataSource dataSource) throws Exception {

SqlSessionFactoryBean factoryBean = new SqlSessionFactoryBean();

factoryBean.setDataSource(dataSource);

return factoryBean.getObject();

}

}

* 扫描service层

@Configuration

@ComponentScan("com.hbnu.service")

public class SpringServiceConfig {

}

* 配置springMVC ，配置jsp访问路径

@Configuration

@ComponentScan("com.hbnu.controller")

@EnableWebMvc

public class SpringWebConfig implements WebMvcConfigurer {

@Override

public void configureDefaultServletHandling(DefaultServletHandlerConfigurer configurer) {

configurer.enable();

}

@Override

public void configureViewResolvers(ViewResolverRegistry registry) {

registry.jsp("/WEB-INF/pages/", ".jsp");

}

}

* 使用注解配置springMVC的DispatcherServlet

public class WebInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {

@Override

protected Class<?>[] getRootConfigClasses() {

return new Class[]{SpringRepositoryConfig.class, SpringServiceConfig.class};

}

@Override

protected Class<?>[] getServletConfigClasses() {

return new Class[]{SpringWebConfig.class};

}

@Override

protected String[] getServletMappings() {

return new String[]{"/"};

}

}

* Controller写法

@RestController //返回json格式

@RequestMapping("/goods") //请求路径

public class GoodsController {

@Resource //自动注入

private GoodsService goodsService;

@RequestMapping("/findAllGoods") //请求路径 浏览器访问/goods/findAllGoods就可以访问到这里

public List<Goods> findAllGoods() {

return goodsService.findAllGoods();

}

}

* Service层写法

@Service

public class GoodsServiceImpl implements GoodsService {

// @Autowired

@Resource

private GoodsDao goodsDao;

@Override

public List<Goods> findAllGoods() {

return goodsDao.findAllGoods();

}

}

* Dao层写法

public interface GoodsDao {

@Select("select \* from tb\_goods")

public List<Goods> findAllGoods();

}

### Ssm xml配置

* Mybatis xml写sql

public interface GoodsMapper {

List<Goods> findAllGoods();

}

<?xml version="1.0" encoding="utf-8" ?>

<!DOCTYPE mapper

PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"

"https://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="com.hbnu.dao.GoodsMapper">

<select id="findAllGoods" resultType="com.hbnu.pojo.Goods">

select \* from tb\_goods

</select>

</mapper>

* Springmvc配置

<?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:mvc="http://www.springframework.org/schema/mvc"

xmlns:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="http://www.springframework.org/schema/mvc

http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd

http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-4.0.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-4.0.xsd">

<!-- 1、放行静态资源配置 -->

<mvc:default-servlet-handler />

<!-- 2、注解识别 -->

<mvc:annotation-driven>

<mvc:message-converters>

<bean class="org.springframework.http.converter.StringHttpMessageConverter">

<property name="supportedMediaTypes">

<list>

<value>text/plain;charset=utf-8</value>

<value>text/html;charset=UTF-8</value>

</list>

</property>

</bean>

</mvc:message-converters>

</mvc:annotation-driven>

<!-- 3、内部视图解析器 -->

<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="prefix" value="/WEB-INF/pages/" />

<property name="suffix" value=".jsp" />

</bean>

<!-- 4、包扫描 -->

<context:component-scan base-package="com.hbnu.controller" />

</beans>

* 配置mybatis和service包扫描

<?xml version="1.0" encoding="utf-8" ?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:context="http://www.springframework.org/schema/context"

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-4.0.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-4.0.xsd">

<!-- 配置数据源dataSource -->

<bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">

<property name="url" value="jdbc:mysql://localhost:3306/class2013?serverTimezone=GMT&amp;useSSL=false"/>

<property name="username" value="root"/>

<property name="password" value="chendikai"/>

</bean>

<!-- 配置SqlSessionFactory -->

<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">

<property name="dataSource" ref="dataSource"/>

<property name="mapperLocations" value="classpath:mybatis/\*.xml"/>

</bean>

<!-- mapper接口扫描 -->

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

<property name="basePackage" value="com.hbnu.dao"/>

</bean>

<!-- service层包扫描 -->

<context:component-scan base-package="com.hbnu.service"/>

</beans>

* Web.xml

<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee

http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd"

version="4.0">

<filter>

<filter-name>encodingFilter</filter-name>

<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>

<init-param>

<param-name>encoding</param-name>

<param-value>utf-8</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>encodingFilter</filter-name>

<url-pattern>/</url-pattern>

</filter-mapping>

<servlet>

<servlet-name>springmvc</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:spring/\*.xml</param-value>

</init-param>

</servlet>

<servlet-mapping>

<servlet-name>springmvc</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

</web-app>