# 整合步骤

## Web.xml

<!-- 配置过滤器，解决hibernate延迟加载问题 -->

<filter>

<filter-name>openSessionInView</filter-name>

<filter-class>org.springframework.orm.hibernate5.support.OpenSessionInViewFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>openSessionInView</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<!-- 加载spring容器 -->

<context-param>

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

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

</context-param>

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

## applicationContext-dao.cml

<!--第一步 加载properties配置文件 -->

<context:property-placeholder location="classpath:config/db.properties"/>

<!--第二步 组件扫描 -->

<context:component-scan base-package="com.wwl.bos"/>

<!--第三步 配置数据源 -->

<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">

<property name="driverClass" value="${jdbc.driverClass}"/>

<property name="jdbcUrl" value="${jdbc.jdbcUrl}"/>

<property name="user" value="${jdbc.user}"/>

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

</bean>

<!--第四步 配置LocalSessionFactoryBean，spring提供的用于整合hibernate的工厂bean -->

<bean id="sessionFactory" class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">

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

<!--这里把hibernate核心配置文件干掉:在这里注入hibernate相关的属性配置 -->

<property name="hibernateProperties">

<props>

<!-- 方言：mysql -->

<prop key="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</prop>

<!-- 自动创建sql表 值有好几种：update：最常用的属性，第一次加载hibernate时根据model类会自动建立起表的结构（前提是先建立好数据库） -->

<prop key="hibernate.hbm2ddl.auto">update</prop>

<!-- 控制台打印sql -->

<prop key="hibernate.show\_sql">true</prop>

<!-- sql格式化 -->

<prop key="hibernate.format\_sql">true</prop>

</props>

</property>

<!--第五步 注入hibernate的映射文件 -->

<property name="mappingLocations">

<list>

<value>classpath:com/wwl/bos/domain/\*.xml</value>

</list>

</property>

</bean>

<!--第六步 配置事务管理器 -->

<bean id="transactionManager" class="org.springframework.orm.hibernate5.HibernateTransactionManager">

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

</bean>

<!-- 使用注解事务 -->

<tx:annotation-driven/>

## Dao

这里接口类就不写了

// 继承模板spring封装的模板HibernateDaoSupport类实现底层BaseDao

public class BaseDaoImpl<T> extends HibernateDaoSupport implements BaseDao<T> {

private Class<T> entityClass;

@Resource // 根据类型注入spring工厂中的会话工厂对象sessionFactory

public void setMySessionFactory(SessionFactory sessionFactory) {

super.setSessionFactory(sessionFactory);

}

// 在父类（BaseDaoImpl）的构造方法中动态获得entityClass

public BaseDaoImpl() {

// 如果子类初始化那么当前this就是子类：通过子类获得父类

ParameterizedType superclass = (ParameterizedType) this.getClass().getGenericSuperclass();

// 获得父类上声明的泛型数组：泛型本身可以声明多个：<T,E,c>

Type[] actualTypeArguments = superclass.getActualTypeArguments();

entityClass = (Class<T>) actualTypeArguments[0];

}

public void save(T entity) { //保存

this.getHibernateTemplate().save(entity);

}

public void delete(T entity) { //删除

this.getHibernateTemplate().delete(entity);

}

public void update(T entity) { //更新

this.getHibernateTemplate().update(entity);

}

public T findById(Serializable id) { //根据id查询

return this.getHibernateTemplate().get(entityClass, id);

}

public List<T> findAll() { //查询所有

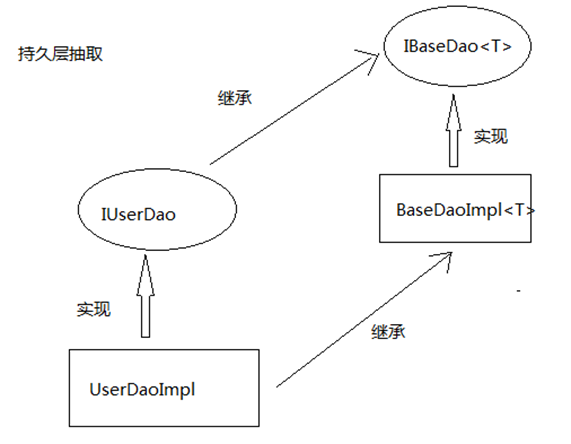
String hql = "FROM " + entityClass.getSimpleName();

return (List<T>) this.getHibernateTemplate().find(hql);

}

}

使用原理图



# 分页操作

第一步：使用easyUI分页传入两个参数：页数和每页总数

请求网址:http://127.0.0.1:8848/page/show?page=1&rows=10

第二步：使用实体类PageBean封装数据

PageBean实体类的成员变量如下：

private int currentPage;//当前页码—》page

private int pageSize;//每页显示的记录数—》rows

private DetachedCriteria detachedCriteria;//查询条件

private int total;//总记录数

private List rows;//当前页需要展示的数据集合

具体封装操作如下：

PageBean pageBean = new PageBean();

pageBean.setCurrentPage(page);

pageBean.setPageSize(rows);

//创建离线提交查询对象，也就是指定要查询的表

DetachedCriteria detachedCriteria = DetachedCriteria.forClass(Staff.class);

pageBean.setDetachedCriteria(detachedCriteria);

Dao中操作

通用分页查询方法

public void pageQuery(PageBean pageBean) {

int currentPage = pageBean.getCurrentPage();

int pageSize = pageBean.getPageSize();

DetachedCriteria detachedCriteria = pageBean.getDetachedCriteria();

//查询total---总数据量的操作

detachedCriteria.setProjection(Projections.rowCount());//指定hibernate框架发出sql的形式----》select count(\*) from bc\_staff;

List<Long> countList = (List<Long>) this.getHibernateTemplate().findByCriteria(detachedCriteria);

Long count = countList.get(0);

pageBean.setTotal(count.intValue());

//查询rows---当前页需要展示的数据集合，上面设置了只查询表的总数条件，这里设置为null，就是查询表的所有数据

detachedCriteria.setProjection(null);//指定hibernate框架发出sql的形式----》select \* from bc\_staff;

int firstResult = (currentPage - 1) \* pageSize;

int maxResults = pageSize;

//分页查询操作：参数：离线查询类、从第几条开始查、查询多少条

List rows = this.getHibernateTemplate().findByCriteria(detachedCriteria, firstResult, maxResults);

pageBean.setRows(rows);

}

# 条件查询

根据用户名和密码查询用户

public User findUserByUsernameAndPassword(String username, String password) {

String hql = "FROM User u WHERE u.username = ? AND u.password = ?";

List<User> list = (List<User>) this.getHibernateTemplate().find(hql, username,password);

if(list != null && list.size() > 0){

return list.get(0);

}

return null;

}