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

<web-app version=*"2.4"*

xmlns=*"http://java.sun.com/xml/ns/j2ee"*

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

xsi:schemaLocation=*"http://java.sun.com/xml/ns/j2ee*

*http://java.sun.com/xml/ns/j2ee/web-app\_2\_4.xsd"*>

<context-param>

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

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

</context-param>

<!--

<servlet>

<servlet-name>context</servlet-name>

<servlet-class>org.springframework.web.context.ContextLoaderServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

-->

<servlet>

<servlet-name>hsoft</servlet-name>

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

<load-on-startup>2</load-on-startup>

</servlet>

<servlet>

<servlet-name>dwr-invoker</servlet-name>

<servlet-class>org.directwebremoting.servlet.DwrServlet</servlet-class>

<init-param>

<param-name>debug</param-name>

<param-value>true</param-value>

</init-param>

<init-param>

<param-name>pollAndCometEnabled</param-name>

<param-value>true</param-value>

</init-param>

<init-param>

<param-name>logLevel</param-name>

<param-value>ERROR</param-value>

</init-param>

<init-param>

<param-name>allowGetForSafariButMakeForgeryEasier</param-name>

<param-value>true</param-value>

</init-param>

<load-on-startup>4</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>hsoft</servlet-name>

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

</servlet-mapping>

<servlet-mapping>

<servlet-name>dwr-invoker</servlet-name>

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

</servlet-mapping>

<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>

<!-- 使用监听器 -->

<listener>

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

</listener>

<!-- 配置Session监听器 -->

<listener>

<listener-class>com.hsoft.framework.util.SessionCounter</listener-class>

</listener>

<!--<jsp-config>

<taglib>

<taglib-uri>http://java.sun.com/jsp/jstl/core</taglib-uri>

<taglib-location>/WEB-INF/tld/c.tld</taglib-location>

</taglib>

</jsp-config> -->

<session-config>

<session-timeout>30</session-timeout>

</session-config>

<!-- 捕获404 -->

<error-page>

<error-code>404</error-code>

<location>/WEB-INF/jsp/404.jsp</location>

</error-page>

<!-- 捕获500 -->

<error-page>

<error-code>500</error-code>

<location>/WEB-INF/jsp/500.jsp</location>

</error-page>

<welcome-file-list>

<welcome-file>index.jsp</welcome-file>

</welcome-file-list>

</web-app>

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

<property name=*"viewClass"* value=*"org.springframework.web.servlet.view.JstlView"*/>

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

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

</bean>

<!-- 映射 -->

<bean id=*"handlerMapping"* class=*"org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping"*>

<property name=*"interceptors"*>

<list>

<bean class=*"com.hsoft.framework.interceptor.AuthorityInterceptor"* />

</list>

</property>

</bean>

<bean id=*"paramResolver"* class=*"org.springframework.web.servlet.mvc.multiaction.ParameterMethodNameResolver"*>

<property name=*"paramName"*>

<value>method</value>

</property>

</bean>

<!-- 限制上传文件大小 -->

<bean id=*"multipartResolver"* class=*"org.springframework.web.multipart.commons.CommonsMultipartResolver"*>

<property name=*"maxUploadSize"* value=*"100010485761"* />

</bean>

<!-- 异常处理机制 --><!--

<bean id="exceptionResolver" class="org.springframework.web.servlet.handler.SimpleMappingExceptionResolver">

<property name="exceptionMappings">

<props>

<prop key="java.lang.Exception">freindlyError</prop>

</props>

</property>

</bean>-->

<bean name=*"/main.do"* class=*"com.hsoft.framework.ctrl.MainCtrl"*>

<property name=*"methodNameResolver"*><ref bean=*"paramResolver"*/></property>

<property name=*"sysparaManager"* ref=*"sysparaManager"*/>

<property name=*"toolsManager"* ref=*"toolsManager"*/>

<property name=*"mainManager"* ref=*"mainManager"*/>

</bean>

<!-- 验证 -->

<bean id=*"validatorFactory"* class=*"org.springmodules.commons.validator.DefaultValidatorFactory"*>

<property name=*"validationConfigLocations"*>

<list>

<value>WEB-INF/classes/validator/validation.xml</value>

<value>WEB-INF/classes/validator/validator-rules.xml</value>

<value>WEB-INF/classes/validator/custom-validator-rules.xml</value>

</list>

</property>

</bean>

<bean id=*"beanValidator"* class=*"org.springmodules.commons.validator.DefaultBeanValidator"*>

<property name=*"validatorFactory"* ref=*"validatorFactory"*/>

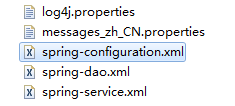
</bean>

<bean id=*"messageSource"* class=*"org.springframework.context.support.ResourceBundleMessageSource"*>

<property name=*"basename"* value=*"messages\_zh\_CN"* />

</bean>

<!-- Validator接口 -->



<!-- 创建数据源与数据库链接 -->

<bean id=*"dataSource"* class=*"org.apache.commons.dbcp.BasicDataSource"* destroy-method=*"close"*>

<property name=*"driverClassName"*><value>oracle.jdbc.driver.OracleDriver</value></property>

<property name=*"url"*><value>jdbc:oracle:thin:@192.168.20.158:1521:orcl</value></property>

<property name=*"username"*><value>cjb</value></property>

<property name=*"password"*><value>cjb</value></property>

<property name=*"maxActive"*><value>20</value></property>

<property name=*"maxIdle"*><value>1</value></property>

<property name=*"maxWait"*><value>10</value></property>

<property name=*"removeAbandoned"*><value>false</value></property>

<property name=*"removeAbandonedTimeout"*><value>30</value></property>

<property name=*"testOnBorrow"*><value>true</value></property>

<property name=*"logAbandoned"*><value>false</value></property>

</bean>

<!-- 事务管理 -->

<bean id=*"transactionManager"* class=*"org.springframework.jdbc.datasource.DataSourceTransactionManager"*>

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

</bean>

<!-- 配置 JDBC模板 -->

<bean id=*"jdbcTemplate"* class=*"com.hsoft.framework.util.FrmJdbcTemplate"*>

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

</bean>

<!-- <bean id="abstractTxDefinition" class="org.springframework.transaction.interceptor.TransactionProxyFactoryBean" lazy-init="true">

<property name="transactionManager"><ref bean="transactionManager"/></property>

<property name="transactionAttributes">

<props>

<prop key="\*">PROPAGATION\_REQUIRED,readOnly</prop>

<prop key="save\*">PROPAGATION\_REQUIRED</prop>

<prop key="add\*">PROPAGATION\_REQUIRED</prop>

<prop key="update\*">PROPAGATION\_REQUIRED</prop>

<prop key="remove\*">PROPAGATION\_REQUIRED</prop>

<prop key="del\*">PROPAGATION\_REQUIRED</prop>

</props>

</property>

</bean>-->

<!-- 配置事务的传播性 -->

<tx:advice id=*"txAdvice"* transaction-manager=*"transactionManager"*>

<tx:attributes>

<tx:method name=*"save\*"* propagation=*"REQUIRED"* />

<tx:method name=*"update\*"* propagation=*"REQUIRED"* />

<tx:method name=*"del\*"* propagation=*"REQUIRED"* />

<tx:method name=*"remove\*"* propagation=*"REQUIRED"* />

<tx:method name=*"add\*"* propagation=*"REQUIRED"* />

<tx:method name=*"\*"* read-only=*"true"* />

</tx:attributes>

</tx:advice>

<!-- 配置哪些类的哪些方法参与事务 -->

<aop:config>

<aop:pointcut id=*"allServiceMethod"* expression=*"execution(\* com.hsoft.framework.service.impl.\*.\*(..))"*/>

<aop:advisor pointcut-ref=*"allServiceMethod"* advice-ref=*"txAdvice"*/>

</aop:config>

<aop:config>

<aop:pointcut id=*"allServiceMethod1"* expression=*"execution(\* com.hsoft.yyzx.service.imp.\*.\*(..))"*/>

<aop:advisor pointcut-ref=*"allServiceMethod1"* advice-ref=*"txAdvice"*/>

</aop:config>

Spring-dao

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

<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN 2.0//EN" "http://www.springframework.org/dtd/spring-beans-2.0.dtd">

<beans>

<!-- 一、对service bean进行声明性事务管理，具体的service bean的定义在第二部分 -->

<bean id=*"nativeJdbcExtractor"* class=*"org.springframework.jdbc.support.nativejdbc.CommonsDbcpNativeJdbcExtractor"* lazy-init=*"true"* />

<bean id=*"oracleLobHandler"* class=*"org.springframework.jdbc.support.lob.OracleLobHandler"* lazy-init=*"true"*>

<property name=*"nativeJdbcExtractor"* ref=*"nativeJdbcExtractor"* />

</bean>

<bean id=*"sysuserDao"* class=*"com.hsoft.framework.dao.jdbc.SysuserDaoJdbc"*>

<property name=*"jdbcTemplate"*><ref bean=*"jdbcTemplate"*/></property>

</bean>

Spring-service

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

<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN 2.0//EN" "http://www.springframework.org/dtd/spring-beans-2.0.dtd">

<beans>

<!-- 一、对service bean进行声明性事务管理，具体的service bean的定义在第二部分 -->

<!-- <bean id="sysuserManager" parent="abstractTxDefinition">

<property name="target"><ref local="sysuserManagerTarget"/></property>

</bean>-->

<bean id=*"toolsManager"* class=*"com.hsoft.framework.util.ToolsManager"*>

<property name=*"toolsDaoJdbc"*><ref bean=*"toolsDao"*/></property>

<property name=*"codeDao"*><ref bean=*"codeDao"*/></property>

<property name=*"departmentDao"*><ref bean=*"departmentDao"*/></property>

<property name=*"programFoldDao"*><ref bean=*"programFoldDao"*/></property>

<property name=*"sysuserDao"*><ref bean=*"sysuserDao"*/></property>

<property name=*"logDao"*><ref bean=*"logDao"*/></property>

</bean>

<bean id=*"sysuserManagerTarget"* class=*"com.hsoft.framework.service.impl.SysuserManagerImpl"*>

<property name=*"sysuserDao"*><ref bean=*"sysuserDao"*/></property>

<property name=*"departmentDao"*><ref bean=*"departmentDao"*/></property>

<property name=*"toolsDaoJdbc"*><ref bean=*"toolsDao"*/></property>

</bean>

/\* @(#)FrmJdbcTemplate.java

\* 创建日期：2011-04-19

\* Copyright(c)2010~2011 hitsoftware co.,Ltd

\*/

package com.hsoft.framework.util;

import java.util.List;

//import org.springframework.dao.DataAccessException;

import org.springframework.jdbc.core.JdbcTemplate;

/\*\*

\* 执行sql查询语句，返回数据集Object 或List,并通过FrmRowMapper对数据进行规范化处理

\* @see JdbcTemplate

\* @see DataAccessException

\* 更新日志： 修改人 版本号 时间 修改内容 赵敏 1.0.0 2011-04-20 添加注释

\* @author 赵敏

\* @version 1.0.0 2011-04-20

\*/

public class FrmJdbcTemplate extends JdbcTemplate {

/\*\*

\* 将sql查询结果集放入 FrmRowMapper处理后返回Object,query a data-bean by sql

\* @param sql 查询语句

\* @param elementType 数据存储对象

\* @return Object 处理后的结果集对象

\*/

@SuppressWarnings("unchecked")

public Object queryForBean(String sql, Class elementType)

//throws DataAccessException

{

try {

return queryForObject(sql, new FrmRowMapper(elementType));

} catch (Exception ex) {

return null;

}

}

/\*\*

\* 将sql查询结果集放入 FrmRowMapper处理后返回List,query one's list by sql

\* @param sql 查询语句

\* @param elementType 数据存储对象

\* @return List

\*/

public List queryForList(String sql, Class elementType)

//throws DataAccessException

{

return query(sql, new FrmRowMapper(elementType));

}

}

/\* @(#)FrmJdbcTemplate.java

\* 创建日期：2011-04-19

\* Copyright(c)2010~2011 hitsoftware co.,Ltd

\*/

package com.hsoft.framework.util;

import java.lang.reflect.Method;

import java.sql.ResultSet;

import java.sql.ResultSetMetaData;

import java.sql.SQLException;

import java.text.SimpleDateFormat;

import java.util.Date;

import org.springframework.jdbc.core.RowMapper;

/\*\*

\* 对传入的数据集做数据规范化处理

\* @see ResultSet

\* @see ResultSetMetaData

\* @see SimpleDateFormat

\* @see RowMapper

\* 更新日志： 修改人 版本号 时间 修改内容 赵敏 1.0.0 2011-04-20 添加注释

\* @author 赵敏

\* @version 1.0.0 2011-04-20

\*/

public class FrmRowMapper implements RowMapper {

private static Class[] longType = new Class[] { Long.class };

private static Class[] stringType = new Class[] { String.class };

private static Class[] doubleType = new Class[] { Double.class };

private static Class[] blobType = new Class[] { java.sql.Blob.class };

private Class<?> requiredType;

/\*\*

\* 构造函数

\*/

public FrmRowMapper() {

}

/\*\*

\* 构造函数,初始化requiredType

\* @param requiredType 初始化数据

\*/

public FrmRowMapper(Class<?> requiredType) {

this.requiredType = requiredType;

}

/\*\*

\* 对resultSet中的数据进行类型判断后做格式化处理

\* @param resultSet 数据集

\* @param rowNum 行数

\* @return Object

\*/

public Object mapRow(ResultSet resultSet,int rowNum) throws SQLException {

Object objIns = null;

try {

Class<? extends Object> classObject = null;

objIns = requiredType.newInstance();

classObject = objIns.getClass();

ResultSetMetaData rsmd =resultSet.getMetaData();

String colType;

String colName;

Object colValue = null;

Class[] type = null;

for (int i = 1; i <= rsmd.getColumnCount(); i++) {

colName = formatColName(rsmd.getColumnName(i));

colType = rsmd.getColumnTypeName(i);

if (!colName.equals("Rowcounter")) {

if (colType.equals("CHAR") || colType.equals("VARCHAR2")||colType.equals("ROWID")) {

colValue =resultSet.getString(colName);

type = stringType;

if (colValue == null)

colValue = "";

} else if (colType.equals("DATE")) {

colValue = cDateStrWithSecond(resultSet.getTimestamp(colName));

type = stringType;

if (colValue == null)

colValue = "";

} else if (colType.equals("NUMBER")) {

if (rsmd.getScale(i) <= 0) {

if (rsmd.getPrecision(i) <= 0) {

if (resultSet.getString(colName) != null) {

colValue = new Double(resultSet.getDouble(colName));

} else {

colValue = null;

}

type = doubleType;

}else{

if (resultSet.getString(colName) != null) {

colValue = Long.valueOf(resultSet.getLong(colName));

} else {

colValue = null;

}

type = longType;

}

} else {

if (resultSet.getString(colName) != null) {

colValue = new Double(resultSet.getDouble(colName));

} else {

colValue = null;

}

type = doubleType;

}

} else if (colType.equals("BLOB")) {

colValue = resultSet.getBlob(colName);

type = blobType;

}

Object[] values={colValue};

//System.out.println("替换前colName::::"+colName);

colName = StringUtil.replaceUnderlineAndfirstToUpper(colName,"\_","");

//System.out.println("替换后colName::::"+colName);

Method meth = classObject.getMethod("set" + colName, type);

meth.invoke(objIns, values);

}

}

}

catch (SQLException ex){

ex.printStackTrace();

}

catch (Exception e) {

e.printStackTrace();

}

return objIns;

}

/\*\*

\* 将指定字符串的第一个字母转换为大写，其它为小写

\* @param colName 需要格式化的字符串

\* @return String

\*/

private static String formatColName(String colName) {

return colName.substring(0, 1).toUpperCase() + colName.substring(1).toLowerCase();

}

/\*\*

\* 格式化传入日期nDate为yyyy-MM-dd HH:mm:ss

\* @param nDate 需要格式化的日期

\* @return String

\*/

protected static String cDateStrWithSecond(Date nDate) {

if (nDate == null) {

return "";

} else {

SimpleDateFormat d = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

return d.format(nDate);

}

}

// public Object mapRow(ResultSet rs, int rowNum) throws SQLException {

// Object objIns = null;

// try {

// objIns = requiredType.newInstance();

// ResultSetMetaData rsmd = rs.getMetaData();

// int columnCount = rsmd.getColumnCount();

// Map mapOfColValues = createColumnMap(columnCount);

// for (int i = 1; i <= columnCount; i++) {

// String key = getColumnKey(rsmd.getColumnName(i));

// Object obj = getColumnValue(rs, i);

// mapOfColValues.put(key, obj);

// }

// BeanUtils.populate(objIns, mapOfColValues);

// } catch (Exception ex) {

// ex.printStackTrace();

// }

// return objIns;

// }

//

// protected Map createColumnMap(int columnCount) {

// return CollectionFactory.createLinkedCaseInsensitiveMapIfPossible(columnCount);

// }

//

// protected String getColumnKey(String columnName) {

// return columnName.toLowerCase();

// }

//

// protected Object getColumnValue(ResultSet rs, int index) throws SQLException {

// return JdbcUtils.getResultSetValue(rs, index);

// }

}

### direct log messages to stdout ###

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.Target=System.out

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%d{ABSOLUTE} %5p %c**{1}**:%L - %m%n

### direct messages to file hibernate.log ###

log4j.appender.file=org.apache.log4j.FileAppender

log4j.appender.file.File=D\:\\logtest.log

log4j.appender.file.layout=org.apache.log4j.PatternLayout

log4j.appender.file.layout.ConversionPattern=%d{ABSOLUTE} %5p %c**{1}**:%L - %m%n

### set log levels - for more verbose logging change 'info' to 'debug' ###

log4j.rootLogger=warn, stdout ,file

#log4j.logger.org.hibernate=info

#è®¾ç½®com.tianyitime.log.serviceåä¸ç±»çä¿¡æ¯è¾åºç­çº§

log4j.logger.com.hsoft.yyzx=info

log4j.logger.com.hsoft=info

#\u8bbe\u7f6ecom.tianyitime.log.service\u5305\u4e0b\u7c7b\u7684\u4fe1\u606f\u8f93\u51fa\u7b49\u7ea7

#log4j.logger.com.hsoft.framework.action=error,warn