宝时WCS源码文档

项目架构

组件应用

<?xml version="1.0" encoding="UTF-8"?>	
	<project xmlns="http://maven.apache.org/POM/4.0.0"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<groupId>com.baoshi</groupId>
	<artifactId>wcs</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<packaging>jar</packaging>
	<name>wcs</name>
	<description>宝时仓储机器人</description>
	<parent>
	<groupId>org.springframework.boot</groupId>
	<artifactId>spring-boot-starter-parent</artifactId>
	<version>2.0.6.RELEASE</version>
	<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<properties>
	<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
	<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
	<java.version>1.8</java.version>
	</properties>

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-jdbc</artifactId>
    </dependency>
    <!--<dependency>-->
    <!--<groupId>org.springframework.boot</groupId>-->
    <!--<artifactId>spring-boot-starter-logging</artifactId>-->
    <!--</dependency>-->
    <!--热启动：每自修改后， 程序自动启动spring Application上下文。 -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-aop</artifactId>
    </dependency>
    <dependency>
        <groupId>org.mybatis.spring.boot</groupId>
        <artifactId>mybatis-spring-boot-starter</artifactId>
        <version>1.3.2</version>
    </dependency>
    <dependency>
        <groupId>com.baomidou</groupId>
        <artifactId>mybatis-plus-boot-starter</artifactId>
        <version>3.0.5</version>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <scope>runtime</scope>
    </dependency>
```

```
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>

        <!-- https://mvnrepository.com/artifact/com.alibaba/fastjson -->
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>fastjson</artifactId>
            <version>1.2.51</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-validation</artifactId>
        </dependency>

        <!-- https://mvnrepository.com/artifact/org.freemarker/freemarker -->
        <dependency>
            <groupId>org.freemarker</groupId>
            <artifactId>freemarker</artifactId>
            <version>2.3.28</version>
        </dependency>

        <!--<dependency>-->
        <!--<groupId>org.springframework.boot</groupId>-->
        <!--<artifactId>spring-boot-starter-security</artifactId>-->
        <!--</dependency>-->

        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger2</artifactId>
            <version>2.9.2</version>
        </dependency>
        <dependency>
```

```
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger-ui</artifactId>
            <version>2.9.2</version>
        </dependency>
    </dependencies>
    <build>
        <resources>
            <resource>
                <!-- xml放在java目录下-->
                <directory>src/main/java</directory>
                <includes>
                    <include>**/*.xml</include>
                </includes>
            </resource>
            <!--指定资源的位置（xml放在resources下，可以不用指定）-->
            <resource>
                <directory>src/main/resources</directory>
            </resource>
        </resources>
        <plugins>
            <!-- 打可执行jar的配置 -->
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <version>1.3.3.RELEASE</version>
                <executions>
                    <execution>
                        <goals>
                            <goal>repackage</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
        </plugins>
        <finalName>wcs</finalName>
    </build>


</project>
```

程序入口

```
package com.baoshi.wcs;

import org.mybatis.spring.annotation.MapperScan;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration;

@SpringBootApplication
@MapperScan("com.baoshi.wcs.dao")
public class WcsApplication {

    public static void main(String[] args) {
        SpringApplication.run(WcsApplication.class, args);
    }
}
```

配置

全局事物 异常捕获 日志存储

```
package com.baoshi.wcs.aspect;

import com.baoshi.wcs.common.enumeration.EventCodeEnum;
import com.baoshi.wcs.common.enumeration.RequestorEnum;
import com.baoshi.wcs.common.response.WCSApiResponse;
import com.baoshi.wcs.common.utils.DateUtil;
import com.baoshi.wcs.entity.Requestor;
import com.baoshi.wcs.service.RequestorService;
import org.aspectj.lang.ProceedingJoinPoint;
import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Pointcut;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpStatus;
import org.springframework.util.StringUtils;
import org.springframework.web.context.request.RequestAttributes;
import org.springframework.web.context.request.RequestContextHolder;
import org.springframework.web.context.request.ServletRequestAttributes;

import javax.servlet.http.HttpServletRequest;
import java.text.ParseException;
import java.util.Date;
```

```
@Aspect
@Configuration
public class LogRecordAspect {
    private static final Logger logger = LoggerFactory.getLogger(LogRecordAspect.class);

    @Autowired
    RequestorService requestorService;

    /**
     * 定义切点Pointcut
     */
    @Pointcut("execution(* com.baoshi.wcs.web.controller.*Controller.*(..))")
    public void excudeService() {
    }

    @Around("excudeService()")
    public Object doAround(ProceedingJoinPoint pjp) throws Throwable {
        RequestAttributes ra = RequestContextHolder.getRequestAttributes();
        ServletRequestAttributes sra = (ServletRequestAttributes) ra;
        HttpServletRequest request = sra.getRequest();
        WCSApiResponse response = new WCSApiResponse();
        Requestor requestor = this.getReqeestorInfo(request);

        // result的值就是被拦截方法的返回值
        Object result = null;
        try {
            result = pjp.proceed();
        } catch (Throwable ex) {
            logger.error("aop ex :", ex);
            if (ex instanceof org.springframework.web.servlet.NoHandlerFoundException) {
                response.setCode(HttpStatus.NOT_FOUND.value());
                response.setMsg("服务路径访问错误.");
            } else {
                response.setCode(HttpStatus.INTERNAL_SERVER_ERROR.value());
                response.setMsg("服务内部错误");
            }
            response.setRequestorVO(requestor);
            response.setServerMsg(ex.getMessage());
            return response;
        }

        boolean saveRes = requestorService.save(requestor);

        if (result instanceof WCSApiResponse) {
            //如果返回的类型是WCSApiResponse 则强转  代替之前声明的WCSApiResponse
            response = (WCSApiResponse) result;

        } else {
```

```
            response.setData(result);
            response.setCode(HttpStatus.OK.value());
        }
        response.setRequestorVO(requestor);
        return response;
    }


    private Requestor getReqeestorInfo(HttpServletRequest request) {
        String requestorCode = request.getParameter("requestor");
        String reqDate = request.getParameter("reqDate");
        String eventCode = request.getParameter("eventCode");

        /**
         * 校验调用者
         */
        if (StringUtils.isEmpty(requestorCode)) {
            requestorCode = Integer.toString(RequestorEnum.GENERAL_USER.getCode());
        } else {

            String role = RequestorEnum.getRole(Integer.parseInt(requestorCode));
            if (StringUtils.isEmpty(role)) {
                //TODO 错误的请求角色编码,暂时处理为普通用户
                requestorCode =
Integer.toString(RequestorEnum.GENERAL_USER.getCode());
            }
        }
        Requestor requestor = new Requestor();
        requestor.setRequestor(requestorCode);
        /**
         * 校验请求时间
         */
        if (StringUtils.isEmpty(reqDate)) {
            requestor.setReqDate(new Date());
        } else {
            Date reqDateParam = null;
            try {
                reqDateParam = DateUtil.format(DateUtil.YYYY_MM_DD_HHMMSS_Formater,
reqDate);
            } catch (ParseException e) {
                reqDateParam = new Date();
            }
            requestor.setReqDate(reqDateParam);
        }

        /**
         * 校验事件编码
         */
        if (StringUtils.isEmpty(eventCode)) {
```

```
            eventCode =
Integer.toString(EventCodeEnum.UNKNOWN_EVENT.getEventCode());
        }
        requestor.setEventCode(eventCode);

        return requestor;
    }

}
```

扫码称重

入口API

```
package com.baoshi.wcs.web.controller;

import com.alibaba.fastjson.JSON;
import com.baoshi.wcs.common.response.WCSApiResponse;
import com.baoshi.wcs.common.utils.Xml2BeanUtil;
import com.baoshi.wcs.common.wms.Order;
import com.baoshi.wcs.common.wms.WMSServiceResponse;
import com.baoshi.wcs.entity.GoodsWeight;
import com.baoshi.wcs.common.response.ApiResponse;
import com.baoshi.wcs.service.GoodsWeightService;
import com.baoshi.wcs.vo.GoodsWeightVO;

import com.baoshi.wcs.web.basic.BaseController;
import org.apache.cxf.endpoint.Client;
import org.apache.cxf.jaxws.endpoint.dynamic.JaxWsDynamicClientFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Controller;
import org.springframework.util.StringUtils;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import java.util.List;
import java.util.concurrent.*;

@Controller
@RequestMapping("/robot")
public class RobotController extends BaseController {

    @Autowired
    private GoodsWeightService goodsWeightService;
```

```
    @Value("${com.wcs.wms.url}")
    private String wmsServiceUrl;

    @Value("${com.wcs.wms.cid}")
    private String wmsServiceCid;

    @Value("${com.wcs.wms.pwd}")
    private String wmsServicePwd;

    @Value("${com.wcs.wms.warehouseid}")
    private String wmsServiceWarehouseId;

    @Value("${com.wcs.wms.unit}")
    private String wmsServiceUnit;

    private static final String goodsWeightKey =
"2C7FACD3AFC3FFE547FC54CDA076A25D";

    private static ExecutorService executor = new ThreadPoolExecutor(1, 1,
            0L, TimeUnit.MILLISECONDS,
            new LinkedBlockingQueue<Runnable>());
    /**
     *
     * @param goodsWeightVO
     * @return
     */
    @RequestMapping(value = "/goods/weight", method = RequestMethod.POST)
    @ResponseBody
    public Object weigt(@RequestBody GoodsWeightVO goodsWeightVO) throws
InterruptedException, ExecutionException, TimeoutException {
        WCSApiResponse<Boolean> apiResponse = new WCSApiResponse<>();
        if(StringUtils.isEmpty(goodsWeightVO)){
            apiResponse.setCode(HttpStatus.INTERNAL_SERVER_ERROR.value());
            apiResponse.setServerMsg("数据不能为空");
            return apiResponse;
        }
        if(StringUtils.isEmpty(goodsWeightVO.getKey())){
            apiResponse.setCode(HttpStatus.INTERNAL_SERVER_ERROR.value());
            apiResponse.setServerMsg("key不能为空");
            return apiResponse;
        }
        if(! goodsWeightKey.equals(goodsWeightVO.getKey())){
            apiResponse.setCode(HttpStatus.INTERNAL_SERVER_ERROR.value());
            apiResponse.setServerMsg("错误的key值");
            return apiResponse;
        }
        String barCode = goodsWeightVO.getBarCode();
        if(StringUtils.isEmpty(barCode)){
```

```
            apiResponse.setCode(HttpStatus.INTERNAL_SERVER_ERROR.value());
            apiResponse.setServerMsg("barcode 不能为空");
            return apiResponse;
        }
        Double weight = goodsWeightVO.getWeight();
        if(null == weight){
            apiResponse.setCode(HttpStatus.INTERNAL_SERVER_ERROR.value());
            apiResponse.setServerMsg("weight 不能为空");
            return apiResponse;
        }

        WMSServiceResponse<List<Order>> wmsServiceResponse =
checkBarcode2Wms(barCode);
        logger.info("wms 快递单 验证 结果: {}",JSON.toJSONString(wmsServiceResponse));
        String rc = wmsServiceResponse.getRc();
        if(!"0000".equals(rc)){
            apiResponse.setCode(HttpStatus.INTERNAL_SERVER_ERROR.value());
            apiResponse.setServerMsg("验证barcode失败 wms 状态码: "+rc+" msg: "+
wmsServiceResponse.getRm() );
            logger.error("快递单号验证失败:{}",JSON.toJSONString(wmsServiceResponse));
            return apiResponse;
        }

        GoodsWeight goodsWeight = new GoodsWeight();
        goodsWeight.setBarCode(barCode);
        goodsWeight.setWeight(goodsWeightVO.getWeight());
        boolean saveRes = goodsWeightService.save(goodsWeight);

        Callable<String> goodsWeight2WMS = new Callable<String>() {
            @Override
            public String call() throws Exception {
                JaxWsDynamicClientFactory dcf = JaxWsDynamicClientFactory.newInstance();
//                String wsUrl = "http://test3.kucangbao.com/kcb-1.0/cxf/warehouse?wsdl";
                String wsUrl = wmsServiceUrl;
                Client client = dcf.createClient(wsUrl);
                String method = "setOrderWeight";//webservice的方法名
                Object[] result = null;
                String reqXml = "<?xml version=\"1.0\" encoding=\"UTF-8\" ?>\n" +
                        "<setOrderWeight>\n" +
                        "<tid>20140318155513001</tid>\n" +
                        "<cid>" + wmsServiceCid + "</cid>\n" +
                        "<pwd>"+wmsServicePwd+"</pwd>\n" +
                        "<warehouseid>"+wmsServiceWarehouseId+"</warehouseid>\n" +
                        "<sendcode>"+barCode+"</sendcode>\n" +
                        "<weight>"+weight+"</weight>\n" +
                        "<unit>"+wmsServiceUnit+"</unit>\n" +
                        "</setOrderWeight>";
                Object[] res = null;
                try {
```

```
                //调用webservice
                res = client.invoke(method, reqXml);
            } catch (Exception e) {
                e.printStackTrace();
                logger.error("保存 wms 扫码称重 异常 :{}",e);
            }
            return res[0].toString();
        }

    };
    if(saveRes) {

        Future<String> future = executor.submit(goodsWeight2WMS);
        String res = future.get(5, TimeUnit.SECONDS);
        WMSServiceResponse resp = Xml2BeanUtil.getBaseWMSResp(res);
        logger.info("推送快递单号重量返回： {}",JSON.toJSONString(resp));
        if (resp != null) {
            String goodsweihgtRc = resp.getRc();
            if (!"0000".equals(goodsweihgtRc)) {
                apiResponse.setCode(HttpStatus.INTERNAL_SERVER_ERROR.value());
                apiResponse.setServerMsg("保存快递单号 重量失败  wms 状态码: " + rc +
" msg: " + resp.getRm());
                logger.error("快单号-重量推送失败:{}",JSON.toJSONString(resp));
                return apiResponse;
            }
        }
        apiResponse.success(saveRes,"barcode 验证成功,并保存成功, 快递单号和重量成
功推送到WMS");
    }
    return apiResponse;
  }


  private WMSServiceResponse<List<Order>>  checkBarcode2Wms(String barcode){
    JaxWsDynamicClientFactory dcf = JaxWsDynamicClientFactory.newInstance();
//      String wsUrl = "http://demo.kucangbao.com/kcb-1.0/cxf/warehouse?wsdl";

//      String wsUrl = "http://test3.kucangbao.com/kcb-1.0/cxf/warehouse?wsdl";
    String wsUrl = wmsServiceUrl;
    Client client = dcf.createClient(wsUrl);
    String method = "getOrders";//webservice的方法名
    Object[] result = null;
    String reqXml = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>" +
            "<getOrders>\n" +
            "<tid>20140318155513001</tid>\n" +
            "<cid>"+wmsServiceCid+"</cid>\n" +
            "<pwd>"+wmsServicePwd+"</pwd>\n" +
            "<warehouseid>"+wmsServiceWarehouseId+"</warehouseid>\n" +
            "<sendcode>"+barcode+"</sendcode>\n" +
```

```
                "</getOrders>";
        WMSServiceResponse<List<Order>> orderDetails = null;
        try {
            result = client.invoke(method, reqXml);//调用webservice

            orderDetails = Xml2BeanUtil.getOrderDetails(result[0].toString());
            return orderDetails;
        } catch (Exception e) {
            e.printStackTrace();
        }
        return orderDetails;
    }
}
```

业务层(通用)

```
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by Fernflower decompiler)
//

package com.baomidou.mybatisplus.extension.service;

import com.baomidou.mybatisplus.core.conditions.Wrapper;
import com.baomidou.mybatisplus.core.metadata.IPage;
import java.io.Serializable;
import java.util.Collection;
import java.util.List;
import java.util.Map;

public interface IService<T> {
    boolean save(T var1);

    default boolean saveBatch(Collection<T> entityList) {
        return this.saveBatch(entityList, 30);
    }

    boolean saveBatch(Collection<T> var1, int var2);

    boolean saveOrUpdateBatch(Collection<T> var1);

    boolean saveOrUpdateBatch(Collection<T> var1, int var2);

    boolean removeById(Serializable var1);

    boolean removeByMap(Map<String, Object> var1);

    boolean remove(Wrapper<T> var1);
```

```
    boolean removeByIds(Collection<? extends Serializable> var1);

    boolean updateById(T var1);

    boolean update(T var1, Wrapper<T> var2);

    default boolean updateBatchById(Collection<T> entityList) {
        return this.updateBatchById(entityList, 30);
    }

    boolean updateBatchById(Collection<T> var1, int var2);

    boolean saveOrUpdate(T var1);

    T getById(Serializable var1);

    Collection<T> listByIds(Collection<? extends Serializable> var1);

    Collection<T> listByMap(Map<String, Object> var1);

    default T getOne(Wrapper<T> queryWrapper) {
        return this.getOne(queryWrapper, false);
    }

    T getOne(Wrapper<T> var1, boolean var2);

    Map<String, Object> getMap(Wrapper<T> var1);

    Object getObj(Wrapper<T> var1);

    int count(Wrapper<T> var1);

    List<T> list(Wrapper<T> var1);

    IPage<T> page(IPage<T> var1, Wrapper<T> var2);

    List<Map<String, Object>> listMaps(Wrapper<T> var1);

    List<Object> listObjs(Wrapper<T> var1);

    IPage<Map<String, Object>> pageMaps(IPage<T> var1, Wrapper<T> var2);
}
```

业务层(实现)

```
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by Fernflower decompiler)
```

```
//

package com.baomidou.mybatisplus.extension.service.impl;

import com.baomidou.mybatisplus.core.conditions.Wrapper;
import com.baomidou.mybatisplus.core.enums.SqlMethod;
import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.baomidou.mybatisplus.core.metadata.IPage;
import com.baomidou.mybatisplus.core.metadata.TableInfo;
import com.baomidou.mybatisplus.core.toolkit.CollectionUtils;
import com.baomidou.mybatisplus.core.toolkit.ExceptionUtils;
import com.baomidou.mybatisplus.core.toolkit.GlobalConfigUtils;
import com.baomidou.mybatisplus.core.toolkit.ObjectUtils;
import com.baomidou.mybatisplus.core.toolkit.ReflectionKit;
import com.baomidou.mybatisplus.core.toolkit.StringUtils;
import com.baomidou.mybatisplus.core.toolkit.TableInfoHelper;
import com.baomidou.mybatisplus.core.toolkit.sql.SqlHelper;
import com.baomidou.mybatisplus.extension.service.IService;
import java.io.Serializable;
import java.util.Collection;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Objects;
import java.util.function.Predicate;
import java.util.stream.Collectors;
import org.apache.ibatis.binding.MapperMethod.ParamMap;
import org.apache.ibatis.session.SqlSession;
import org.mybatis.spring.SqlSessionUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.transaction.annotation.Transactional;

public class ServiceImpl<M extends BaseMapper<T>, T> implements IService<T> {
    @Autowired
    protected M baseMapper;

    public ServiceImpl() {
    }

    protected boolean retBool(Integer result) {
        return SqlHelper.retBool(result);
    }

    protected Class<T> currentModelClass() {
        return ReflectionKit.getSuperClassGenericType(this.getClass(), 1);
    }

    protected SqlSession sqlSessionBatch() {
        return SqlHelper.sqlSessionBatch(this.currentModelClass());
```

```
    }

    protected void closeSqlSession(SqlSession sqlSession) {
        SqlSessionUtils.closeSqlSession(sqlSession,
GlobalConfigUtils.currentSessionFactory(this.currentModelClass()));
    }

    protected String sqlStatement(SqlMethod sqlMethod) {
        return
SqlHelper.table(this.currentModelClass()).getSqlStatement(sqlMethod.getMethod());
    }

    @Transactional(
        rollbackFor = {Exception.class}
    )
    public boolean save(T entity) {
        return this.retBool(this.baseMapper.insert(entity));
    }

    @Transactional(
        rollbackFor = {Exception.class}
    )
    public boolean saveBatch(Collection<T> entityList, int batchSize) {
        int i = 0;
        String sqlStatement = this.sqlStatement(SqlMethod.INSERT_ONE);
        SqlSession batchSqlSession = this.sqlSessionBatch();
        Throwable var6 = null;

        try {
            for(Iterator var7 = entityList.iterator(); var7.hasNext(); ++i) {
                T anEntityList = var7.next();
                batchSqlSession.insert(sqlStatement, anEntityList);
                if (i >= 1 && i % batchSize == 0) {
                    batchSqlSession.flushStatements();
                }
            }

            batchSqlSession.flushStatements();
            return true;
        } catch (Throwable var16) {
            var6 = var16;
            throw var16;
        } finally {
            if (batchSqlSession != null) {
                if (var6 != null) {
                    try {
                        batchSqlSession.close();
                    } catch (Throwable var15) {
                        var6.addSuppressed(var15);
```

```
                }
            } else {
                batchSqlSession.close();
            }
        }

    }
}

@Transactional(
    rollbackFor = {Exception.class}
)
public boolean saveOrUpdate(T entity) {
    if (null == entity) {
        return false;
    } else {
        Class<?> cls = entity.getClass();
        TableInfo tableInfo = TableInfoHelper.getTableInfo(cls);
        if (null != tableInfo && StringUtils.isNotEmpty(tableInfo.getKeyProperty())) {
            Object idVal = ReflectionKit.getMethodValue(cls, entity,
tableInfo.getKeyProperty());
            if (StringUtils.checkValNull(idVal)) {
                return this.save(entity);
            } else {
                return this.updateById(entity) || this.save(entity);
            }
        } else {
            throw ExceptionUtils.mpe("Error:  Can not execute. Could not find
@TableId.");
        }
    }
}

@Transactional(
    rollbackFor = {Exception.class}
)
public boolean saveOrUpdateBatch(Collection<T> entityList) {
    return this.saveOrUpdateBatch(entityList, 30);
}

@Transactional(
    rollbackFor = {Exception.class}
)
public boolean saveOrUpdateBatch(Collection<T> entityList, int batchSize) {
    if (CollectionUtils.isEmpty(entityList)) {
        throw new IllegalArgumentException("Error: entityList must not be empty");
    } else {
        Class<?> cls = null;
        TableInfo tableInfo = null;
```

```
            int i = 0;
            SqlSession batchSqlSession = this.sqlSessionBatch();
            Throwable var7 = null;

            try {
                Iterator var8 = entityList.iterator();

                while(var8.hasNext()) {
                    T anEntityList = var8.next();
                    if (i == 0) {
                        cls = anEntityList.getClass();
                        tableInfo = TableInfoHelper.getTableInfo(cls);
                    }

                    if (null == tableInfo || !StringUtils.isNotEmpty(tableInfo.getKeyProperty()))
{
                        throw ExceptionUtils.mpe("Error:  Can not execute. Could not find
@TableId.");
                    }

                    Object idVal = ReflectionKit.getMethodValue(cls, anEntityList,
tableInfo.getKeyProperty());
                    String sqlStatement;
                    if (StringUtils.checkValNull(idVal)) {
                        sqlStatement = this.sqlStatement(SqlMethod.INSERT_ONE);
                        batchSqlSession.insert(sqlStatement, anEntityList);
                    } else {
                        sqlStatement = this.sqlStatement(SqlMethod.UPDATE_BY_ID);
                        ParamMap<T> param = new ParamMap();
                        param.put("et", anEntityList);
                        batchSqlSession.update(sqlStatement, param);
                    }

                    if (i >= 1 && i % batchSize == 0) {
                        batchSqlSession.flushStatements();
                    }

                    ++i;
                    batchSqlSession.flushStatements();
                }
            } catch (Throwable var20) {
                var7 = var20;
                throw var20;
            } finally {
                if (batchSqlSession != null) {
                    if (var7 != null) {
                        try {
                            batchSqlSession.close();
                        } catch (Throwable var19) {
```

```
                    var7.addSuppressed(var19);
                }
            } else {
                batchSqlSession.close();
            }
        }

    }

    return true;
  }
}

@Transactional(
    rollbackFor = {Exception.class}
)
public boolean removeById(Serializable id) {
    return SqlHelper.delBool(this.baseMapper.deleteById(id));
}

@Transactional(
    rollbackFor = {Exception.class}
)
public boolean removeByMap(Map<String, Object> columnMap) {
    if (ObjectUtils.isEmpty(columnMap)) {
        throw ExceptionUtils.mpe("removeByMap columnMap is empty.");
    } else {
        return SqlHelper.delBool(this.baseMapper.deleteByMap(columnMap));
    }
}

@Transactional(
    rollbackFor = {Exception.class}
)
public boolean remove(Wrapper<T> wrapper) {
    return SqlHelper.delBool(this.baseMapper.delete(wrapper));
}

@Transactional(
    rollbackFor = {Exception.class}
)
public boolean removeByIds(Collection<? extends Serializable> idList) {
    return SqlHelper.delBool(this.baseMapper.deleteBatchIds(idList));
}

@Transactional(
    rollbackFor = {Exception.class}
)
public boolean updateById(T entity) {
```

```
    return this.retBool(this.baseMapper.updateById(entity));
}

@Transactional(
    rollbackFor = {Exception.class}
)
public boolean update(T entity, Wrapper<T> updateWrapper) {
    return this.retBool(this.baseMapper.update(entity, updateWrapper));
}

@Transactional(
    rollbackFor = {Exception.class}
)
public boolean updateBatchById(Collection<T> entityList, int batchSize) {
    if (CollectionUtils.isEmpty(entityList)) {
        throw new IllegalArgumentException("Error: entityList must not be empty");
    } else {
        int i = 0;
        String sqlStatement = this.sqlStatement(SqlMethod.UPDATE_BY_ID);
        SqlSession batchSqlSession = this.sqlSessionBatch();
        Throwable var6 = null;

        try {
            for(Iterator var7 = entityList.iterator(); var7.hasNext(); ++i) {
                T anEntityList = var7.next();
                ParamMap<T> param = new ParamMap();
                param.put("et", anEntityList);
                batchSqlSession.update(sqlStatement, param);
                if (i >= 1 && i % batchSize == 0) {
                    batchSqlSession.flushStatements();
                }
            }

            batchSqlSession.flushStatements();
            return true;
        } catch (Throwable var17) {
            var6 = var17;
            throw var17;
        } finally {
            if (batchSqlSession != null) {
                if (var6 != null) {
                    try {
                        batchSqlSession.close();
                    } catch (Throwable var16) {
                        var6.addSuppressed(var16);
                    }
                } else {
                    batchSqlSession.close();
                }
```

```
            }

        }
    }
}

public T getById(Serializable id) {
    return this.baseMapper.selectById(id);
}

public Collection<T> listByIds(Collection<? extends Serializable> idList) {
    return this.baseMapper.selectBatchIds(idList);
}

public Collection<T> listByMap(Map<String, Object> columnMap) {
    return this.baseMapper.selectByMap(columnMap);
}

public T getOne(Wrapper<T> queryWrapper, boolean throwEx) {
    return throwEx ? this.baseMapper.selectOne(queryWrapper) :
SqlHelper.getObject(this.baseMapper.selectList(queryWrapper));
}

public Map<String, Object> getMap(Wrapper<T> queryWrapper) {
    return (Map)SqlHelper.getObject(this.baseMapper.selectMaps(queryWrapper));
}

public Object getObj(Wrapper<T> queryWrapper) {
    return SqlHelper.getObject(this.baseMapper.selectObjs(queryWrapper));
}

public int count(Wrapper<T> queryWrapper) {
    return SqlHelper.retCount(this.baseMapper.selectCount(queryWrapper));
}

public List<T> list(Wrapper<T> queryWrapper) {
    return this.baseMapper.selectList(queryWrapper);
}

public IPage<T> page(IPage<T> page, Wrapper<T> queryWrapper) {
    return this.baseMapper.selectPage(page, queryWrapper);
}

public List<Map<String, Object>> listMaps(Wrapper<T> queryWrapper) {
    return this.baseMapper.selectMaps(queryWrapper);
}

public List<Object> listObjs(Wrapper<T> queryWrapper) {
```

```
        return (List)this.baseMapper.selectObjs(queryWrapper).stream().filter(Objects::nonNull).collect(Collectors.toList());
    }

    public IPage<Map<String, Object>> pageMaps(IPage<T> page, Wrapper<T> queryWrapper) {
        return this.baseMapper.selectMapsPage(page, queryWrapper);
    }
}
```

持久层(接口)

```
package com.baoshi.wcs.dao;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.baoshi.wcs.entity.GoodsWeight;

/**
 * <p>
 * 货物/包裹信息表 Mapper 接口
 * </p>
 *
 * @author jobob
 * @since 2018-11-11
 */
public interface GoodsWeightMapper extends BaseMapper<GoodsWeight> {

}
```

持久层(实现)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.baoshi.wcs.dao.GoodsWeightMapper">

    <!-- 通用查询映射结果 -->
    <resultMap id="BaseResultMap" type="com.baoshi.wcs.entity.GoodsWeight">
    <result column="id" property="id" />
    <result column="create_time" property="createTime" />
    <result column="update_time" property="updateTime" />
    <result column="creator" property="creator" />
    <result column="version" property="version" />
        <result column="bar_code" property="barCode" />
        <result column="weight" property="weight" />
        <result column="modifier" property="modifier" />
    </resultMap>
```

```
    <!-- 通用查询结果列 -->
    <sql id="Base_Column_List">
        id,
        create_time,
        update_time,
        creator,
        version,
        bar_code, weight, modifier
    </sql>

</mapper>
```