package org.linlinjava.litemall.core.config;

import org.springframework.context.annotation.Configuration;

import org.springframework.scheduling.annotation.EnableAsync;

@Configuration

@EnableAsync

public class AsyncConfig { }

package org.linlinjava.litemall.core.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.cors.CorsConfiguration;

import org.springframework.web.cors.UrlBasedCorsConfigurationSource;

import org.springframework.web.filter.CorsFilter;

@Configuration

public class CorsConfig {

private CorsConfiguration buildConfig() {

CorsConfiguration corsConfiguration = new CorsConfiguration();

corsConfiguration.addAllowedOrigin("\*"); // 1 设置访问源地址

corsConfiguration.addAllowedHeader("\*"); // 2 设置访问源请求头

corsConfiguration.addAllowedMethod("\*"); // 3 设置访问源请求方法

return corsConfiguration;

}

@Bean

public CorsFilter corsFilter() {

UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();

source.registerCorsConfiguration("/\*\*", buildConfig()); // 4 对接口配置跨域设置

return new CorsFilter(source);

}

}

package org.linlinjava.litemall.core.config;

import org.hibernate.validator.internal.engine.path.PathImpl;

import org.linlinjava.litemall.core.util.ResponseUtil;

import org.springframework.core.Ordered;

import org.springframework.core.annotation.Order;

import org.springframework.http.converter.HttpMessageNotReadableException;

import org.springframework.web.bind.MissingServletRequestParameterException;

import org.springframework.web.bind.annotation.ControllerAdvice;

import org.springframework.web.bind.annotation.ExceptionHandler;

import org.springframework.web.bind.annotation.ResponseBody;

import org.springframework.web.method.annotation.MethodArgumentTypeMismatchException;

import javax.validation.ConstraintViolation;

import javax.validation.ConstraintViolationException;

import javax.validation.ValidationException;

import java.util.Set;

@ControllerAdvice

@Order( value = Ordered.LOWEST\_PRECEDENCE )

public class GlobalExceptionHandler {

@ExceptionHandler(IllegalArgumentException.class)

@ResponseBody

public Object badArgumentHandler(IllegalArgumentException e) {

e.printStackTrace();

return ResponseUtil.badArgumentValue();

}

@ExceptionHandler(MethodArgumentTypeMismatchException.class)

@ResponseBody

public Object badArgumentHandler(MethodArgumentTypeMismatchException e) {

e.printStackTrace();

return ResponseUtil.badArgumentValue();

}

@ExceptionHandler(MissingServletRequestParameterException.class)

@ResponseBody

public Object badArgumentHandler(MissingServletRequestParameterException e) {

e.printStackTrace();

return ResponseUtil.badArgumentValue();

}

@ExceptionHandler(HttpMessageNotReadableException.class)

@ResponseBody

public Object badArgumentHandler(HttpMessageNotReadableException e) {

e.printStackTrace();

return ResponseUtil.badArgumentValue();

}

@ExceptionHandler(ValidationException.class)

@ResponseBody

public Object badArgumentHandler(ValidationException e) {

e.printStackTrace();

if (e instanceof ConstraintViolationException) {

ConstraintViolationException exs = (ConstraintViolationException) e;

Set<ConstraintViolation<?>> violations = exs.getConstraintViolations();

for (ConstraintViolation<?> item : violations) {

String message = ((PathImpl) item.getPropertyPath()).getLeafNode().getName() + item.getMessage();

return ResponseUtil.fail(402, message);

}

}

return ResponseUtil.badArgumentValue();

}

@ExceptionHandler(Exception.class)

@ResponseBody

public Object seriousHandler(Exception e) {

e.printStackTrace();

return ResponseUtil.serious();

}

}

package org.linlinjava.litemall.core.config;

import com.fasterxml.jackson.annotation.JsonInclude;

import com.fasterxml.jackson.databind.SerializationFeature;

import com.fasterxml.jackson.datatype.jsr310.deser.LocalDateDeserializer;

import com.fasterxml.jackson.datatype.jsr310.deser.LocalDateTimeDeserializer;

import com.fasterxml.jackson.datatype.jsr310.deser.LocalTimeDeserializer;

import com.fasterxml.jackson.datatype.jsr310.ser.LocalDateSerializer;

import com.fasterxml.jackson.datatype.jsr310.ser.LocalDateTimeSerializer;

import com.fasterxml.jackson.datatype.jsr310.ser.LocalTimeSerializer;

import org.springframework.boot.autoconfigure.jackson.Jackson2ObjectMapperBuilderCustomizer;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.core.Ordered;

import org.springframework.core.annotation.Order;

import org.springframework.http.converter.json.Jackson2ObjectMapperBuilder;

import java.time.LocalDate;

import java.time.LocalDateTime;

import java.time.LocalTime;

import java.time.format.DateTimeFormatter;

@Configuration

public class JacksonConfig {

@Bean

@Order(Ordered.HIGHEST\_PRECEDENCE)

public Jackson2ObjectMapperBuilderCustomizer customJackson() {

return new Jackson2ObjectMapperBuilderCustomizer() {

@Override

public void customize(Jackson2ObjectMapperBuilder builder) {

builder.serializerByType(LocalDateTime.class, new LocalDateTimeSerializer(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss")));

builder.serializerByType(LocalDate.class, new LocalDateSerializer(DateTimeFormatter.ofPattern("yyyy-MM-dd")));

builder.serializerByType(LocalTime.class, new LocalTimeSerializer(DateTimeFormatter.ofPattern("HH:mm:ss")));

builder.deserializerByType(LocalDateTime.class, new LocalDateTimeDeserializer(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss")));

builder.deserializerByType(LocalDate.class, new LocalDateDeserializer(DateTimeFormatter.ofPattern("yyyy-MM-dd")));

builder.deserializerByType(LocalTime.class, new LocalTimeDeserializer(DateTimeFormatter.ofPattern("HH:mm:ss")));

builder.serializationInclusion(JsonInclude.Include.NON\_NULL);

builder.failOnUnknownProperties(false);

builder.featuresToDisable(SerializationFeature.WRITE\_DATES\_AS\_TIMESTAMPS);

}

};

}

}

package org.linlinjava.litemall.core.config;

import org.hibernate.validator.HibernateValidator;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import javax.validation.Validation;

import javax.validation.Validator;

import javax.validation.ValidatorFactory;

@Configuration

public class ValidatorConfiguration {

@Bean

public Validator validator() {

ValidatorFactory validatorFactory = Validation.byProvider(HibernateValidator.class)

.configure()

.addProperty("hibernate.validator.fail\_fast", "true")

.buildValidatorFactory();

Validator validator = validatorFactory.getValidator();

return validator;

}

}

package org.linlinjava.litemall.core.config;

import cn.binarywang.wx.miniapp.api.WxMaService;

import cn.binarywang.wx.miniapp.api.impl.WxMaServiceImpl;

import cn.binarywang.wx.miniapp.config.WxMaConfig;

import cn.binarywang.wx.miniapp.config.WxMaInMemoryConfig;

import com.github.binarywang.wxpay.config.WxPayConfig;

import com.github.binarywang.wxpay.service.WxPayService;

import com.github.binarywang.wxpay.service.impl.WxPayServiceImpl;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

@Configuration

public class WxConfig {

@Autowired

private WxProperties properties;

@Bean

public WxMaConfig wxMaConfig() {

WxMaInMemoryConfig config = new WxMaInMemoryConfig();

config.setAppid(properties.getAppId());

config.setSecret(properties.getAppSecret());

return config;

}

@Bean

public WxMaService wxMaService(WxMaConfig maConfig) {

WxMaService service = new WxMaServiceImpl();

service.setWxMaConfig(maConfig);

return service;

}

@Bean

public WxPayConfig wxPayConfig() {

WxPayConfig payConfig = new WxPayConfig();

payConfig.setAppId(properties.getAppId());

payConfig.setMchId(properties.getMchId());

payConfig.setMchKey(properties.getMchKey());

payConfig.setNotifyUrl(properties.getNotifyUrl());

payConfig.setKeyPath(properties.getKeyPath());

payConfig.setTradeType("JSAPI");

payConfig.setSignType("MD5");

return payConfig;

}

@Bean

public WxPayService wxPayService(WxPayConfig payConfig) {

WxPayService wxPayService = new WxPayServiceImpl();

wxPayService.setConfig(payConfig);

return wxPayService;

}

}

package org.linlinjava.litemall.core.config;

import org.springframework.boot.context.properties.ConfigurationProperties;

import org.springframework.context.annotation.Configuration;

@Configuration

@ConfigurationProperties(prefix = "litemall.wx")

public class WxProperties {

private String appId;

private String appSecret;

private String mchId;

private String mchKey;

private String notifyUrl;

private String keyPath;

public String getNotifyUrl() {

return notifyUrl;

}

public void setNotifyUrl(String notifyUrl) {

this.notifyUrl = notifyUrl;

}

public String getMchKey() {

return mchKey;

}

public void setMchKey(String mchKey) {

this.mchKey = mchKey;

}

public String getAppId() {

return this.appId;

}

public void setAppId(String appId) {

this.appId = appId;

}

public String getAppSecret() {

return appSecret;

}

public void setAppSecret(String appSecret) {

this.appSecret = appSecret;

}

public String getMchId() {

return mchId;

}

public void setMchId(String mchId) {

this.mchId = mchId;

}

public String getKeyPath() {

return keyPath;

}

public void setKeyPath(String keyPath) {

this.keyPath = keyPath;

}

}

package org.linlinjava.litemall.core;

import org.mybatis.spring.annotation.MapperScan;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication(scanBasePackages={"org.linlinjava.litemall.db","org.linlinjava.litemall.core"})

@MapperScan("org.linlinjava.litemall.db.dao")

public class Application {

public static void main(String[] args) {

SpringApplication.run(Application.class, args);

}

}

package org.linlinjava.litemall.core.express.config;

import org.linlinjava.litemall.core.express.ExpressService;

import org.springframework.boot.context.properties.EnableConfigurationProperties;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

@Configuration

@EnableConfigurationProperties(ExpressProperties.class)

public class ExpressAutoConfiguration {

private final ExpressProperties properties;

public ExpressAutoConfiguration(ExpressProperties properties) {

this.properties = properties;

}

@Bean

public ExpressService expressService() {

ExpressService expressService = new ExpressService();

expressService.setProperties(properties);

return expressService;

}

}

package org.linlinjava.litemall.core.express.config;

import org.springframework.boot.context.properties.ConfigurationProperties;

import java.util.ArrayList;

import java.util.List;

import java.util.Map;

@ConfigurationProperties(prefix = "litemall.express")

public class ExpressProperties {

private boolean enable;

private String appId;

private String appKey;

private List<Map<String, String>> vendors = new ArrayList<>();

public boolean isEnable() {

return enable;

}

public void setEnable(boolean enable) {

this.enable = enable;

}

public List<Map<String, String>> getVendors() {

return vendors;

}

public void setVendors(List<Map<String, String>> vendors) {

this.vendors = vendors;

}

public String getAppKey() {

return appKey;

}

public void setAppKey(String appKey) {

this.appKey = appKey;

}

public String getAppId() {

return appId;

}

public void setAppId(String appId) {

this.appId = appId;

}

}

package org.linlinjava.litemall.core.express;

import com.fasterxml.jackson.databind.ObjectMapper;

import org.linlinjava.litemall.core.express.config.ExpressProperties;

import org.linlinjava.litemall.core.express.dao.ExpressInfo;

import org.linlinjava.litemall.core.util.HttpUtil;

import org.springframework.util.Base64Utils;

import java.net.URLEncoder;

import java.security.MessageDigest;

import java.util.HashMap;

import java.util.Map;

/\*\*

\* 物流查询服务

\*

\* 快递鸟即时查询API http://www.kdniao.com/api-track

\*/

public class ExpressService {

//请求url

private String ReqURL = "http://api.kdniao.com/Ebusiness/EbusinessOrderHandle.aspx";

private ExpressProperties properties;

public ExpressProperties getProperties() {

return properties;

}

public void setProperties(ExpressProperties properties) {

this.properties = properties;

}

/\*\*

\* 获取物流供应商名

\*

\* @param vendorCode

\* @return

\*/

public String getVendorName(String vendorCode) {

for (Map<String, String> item : properties.getVendors()) {

if (item.get("code").equals(vendorCode))

return item.get("name");

}

return null;

}

/\*\*

\* 获取物流信息

\*

\* @param expCode

\* @param expNo

\* @return

\*/

public ExpressInfo getExpressInfo(String expCode, String expNo) {

try {

String result = getOrderTracesByJson(expCode, expNo);

ObjectMapper objMap = new ObjectMapper();

ExpressInfo ei = objMap.readValue(result, ExpressInfo.class);

ei.setShipperName(getVendorName(expCode));

return ei;

} catch (Exception e) {

e.printStackTrace();

}

return null;

}

/\*\*

\* Json方式 查询订单物流轨迹

\*

\* @throws Exception

\*/

private String getOrderTracesByJson(String expCode, String expNo) throws Exception {

if (!properties.isEnable()) {

return null;

}

String requestData = "{'OrderCode':'','ShipperCode':'" + expCode + "','LogisticCode':'" + expNo + "'}";

Map<String, String> params = new HashMap<String, String>();

params.put("RequestData", URLEncoder.encode(requestData, "UTF-8"));

params.put("EBusinessID", properties.getAppId());

params.put("RequestType", "1002");

String dataSign = encrypt(requestData, properties.getAppKey(), "UTF-8");

params.put("DataSign", URLEncoder.encode(dataSign, "UTF-8"));

params.put("DataType", "2");

String result = HttpUtil.sendPost(ReqURL, params);

//根据公司业务处理返回的信息......

return result;

}

/\*\*

\* MD5加密

\*

\* @param str 内容

\* @param charset 编码方式

\* @throws Exception

\*/

private String MD5(String str, String charset) throws Exception {

MessageDigest md = MessageDigest.getInstance("MD5");

md.update(str.getBytes(charset));

byte[] result = md.digest();

StringBuffer sb = new StringBuffer(32);

for (int i = 0; i < result.length; i++) {

int val = result[i] & 0xff;

if (val <= 0xf) {

sb.append("0");

}

sb.append(Integer.toHexString(val));

}

return sb.toString().toLowerCase();

}

/\*\*

\* Sign签名生成

\*

\* @param content 内容

\* @param keyValue Appkey

\* @param charset 编码方式

\* @return DataSign签名

\*/

private String encrypt(String content, String keyValue, String charset) {

if (keyValue != null) {

content = content + keyValue;

}

byte[] src = new byte[0];

try {

src = MD5(content, charset).getBytes(charset);

return Base64Utils.encodeToString(src);

} catch (Exception e) {

e.printStackTrace();

}

return null;

}

}

package org.linlinjava.litemall.core.notify.config;

import com.github.qcloudsms.SmsSingleSender;

import org.linlinjava.litemall.core.notify.NotifyService;

import org.linlinjava.litemall.core.notify.TencentSmsSender;

import org.linlinjava.litemall.core.notify.WxTemplateSender;

import org.springframework.boot.context.properties.EnableConfigurationProperties;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.mail.javamail.JavaMailSender;

import org.springframework.mail.javamail.JavaMailSenderImpl;

@Configuration

@EnableConfigurationProperties(NotifyProperties.class)

public class NotifyAutoConfiguration {

private final NotifyProperties properties;

public NotifyAutoConfiguration(NotifyProperties properties) {

this.properties = properties;

}

@Bean

public NotifyService notifyService() {

NotifyService notifyService = new NotifyService();

NotifyProperties.Mail mailConfig = properties.getMail();

if (mailConfig.isEnable()) {

notifyService.setMailSender(mailSender());

notifyService.setSendFrom(mailConfig.getSendfrom());

notifyService.setSendTo(mailConfig.getSendto());

}

NotifyProperties.Sms smsConfig = properties.getSms();

if (smsConfig.isEnable()) {

notifyService.setSmsSender(tencentSmsSender());

notifyService.setSmsTemplate(smsConfig.getTemplate());

}

NotifyProperties.Wx wxConfig = properties.getWx();

if (wxConfig.isEnable()) {

notifyService.setWxTemplateSender(wxTemplateSender());

notifyService.setWxTemplate(wxConfig.getTemplate());

}

return notifyService;

}

@Bean

public JavaMailSender mailSender() {

NotifyProperties.Mail mailConfig = properties.getMail();

JavaMailSenderImpl mailSender = new JavaMailSenderImpl();

mailSender.setHost(mailConfig.getHost());

mailSender.setUsername(mailConfig.getUsername());

mailSender.setPassword(mailConfig.getPassword());

return mailSender;

}

@Bean

public WxTemplateSender wxTemplateSender() {

WxTemplateSender wxTemplateSender = new WxTemplateSender();

return wxTemplateSender;

}

@Bean

public TencentSmsSender tencentSmsSender() {

NotifyProperties.Sms smsConfig = properties.getSms();

TencentSmsSender smsSender = new TencentSmsSender();

smsSender.setSender(new SmsSingleSender(smsConfig.getAppid(), smsConfig.getAppkey()));

return smsSender;

}

}

package org.linlinjava.litemall.core.notify.config;

import org.springframework.boot.context.properties.ConfigurationProperties;

import java.util.ArrayList;

import java.util.List;

import java.util.Map;

@ConfigurationProperties(prefix = "litemall.notify")

public class NotifyProperties {

private Mail mail;

private Sms sms;

private Wx wx;

public Mail getMail() {

return mail;

}

public void setMail(Mail mail) {

this.mail = mail;

}

public Sms getSms() {

return sms;

}

public void setSms(Sms sms) {

this.sms = sms;

}

public Wx getWx() {

return wx;

}

public void setWx(Wx wx) {

this.wx = wx;

}

public static class Mail {

private boolean enable;

private String host;

private String username;

private String password;

private String sendfrom;

private String sendto;

public boolean isEnable() {

return enable;

}

public void setEnable(boolean enable) {

this.enable = enable;

}

public String getHost() {

return host;

}

public void setHost(String host) {

this.host = host;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public String getSendfrom() {

return sendfrom;

}

public void setSendfrom(String sendfrom) {

this.sendfrom = sendfrom;

}

public String getSendto() {

return sendto;

}

public void setSendto(String sendto) {

this.sendto = sendto;

}

}

public static class Sms {

private boolean enable;

private int appid;

private String appkey;

private List<Map<String, String>> template = new ArrayList<>();

public boolean isEnable() {

return enable;

}

public void setEnable(boolean enable) {

this.enable = enable;

}

public int getAppid() {

return appid;

}

public void setAppid(int appid) {

this.appid = appid;

}

public String getAppkey() {

return appkey;

}

public void setAppkey(String appkey) {

this.appkey = appkey;

}

public List<Map<String, String>> getTemplate() {

return template;

}

public void setTemplate(List<Map<String, String>> template) {

this.template = template;

}

}

public static class Wx {

private boolean enable;

private List<Map<String, String>> template = new ArrayList<>();

public boolean isEnable() {

return enable;

}

public void setEnable(boolean enable) {

this.enable = enable;

}

public List<Map<String, String>> getTemplate() {

return template;

}

public void setTemplate(List<Map<String, String>> template) {

this.template = template;

}

}

}

package org.linlinjava.litemall.core.notify;

import org.springframework.mail.MailSender;

import org.springframework.mail.SimpleMailMessage;

import org.springframework.scheduling.annotation.Async;

import java.util.ArrayList;

import java.util.List;

import java.util.Map;

/\*\*

\* 商城通知服务类

\*/

public class NotifyService {

private MailSender mailSender;

private String sendFrom;

private String sendTo;

private SmsSender smsSender;

private List<Map<String, String>> smsTemplate = new ArrayList<>();

private WxTemplateSender wxTemplateSender;

private List<Map<String, String>> wxTemplate = new ArrayList<>();

public boolean isMailEnable() {

return mailSender != null;

}

public boolean isSmsEnable() {

return smsSender != null;

}

public boolean isWxEnable() {

return wxTemplateSender != null;

}

/\*\*

\* 短信消息通知

\*

\* @param phoneNumber 接收通知的电话号码

\* @param message 短消息内容，这里短消息内容必须已经在短信平台审核通过

\*/

@Async

public void notifySms(String phoneNumber, String message) {

if (smsSender == null)

return;

smsSender.send(phoneNumber, message);

}

/\*\*

\* 短信模版消息通知

\*

\* @param phoneNumber 接收通知的电话号码

\* @param notifyType 通知类别，通过该枚举值在配置文件中获取相应的模版ID

\* @param params 通知模版内容里的参数，类似"您的验证码为{1}"中{1}的值

\*/

@Async

public void notifySmsTemplate(String phoneNumber, NotifyType notifyType, String[] params) {

if (smsSender == null) {

return;

}

String templateIdStr = getTemplateId(notifyType, smsTemplate);

if (templateIdStr == null) {

return;

}

int templateId = Integer.parseInt(templateIdStr);

smsSender.sendWithTemplate(phoneNumber, templateId, params);

}

/\*\*

\* 以同步的方式发送短信模版消息通知

\*

\* @param phoneNumber 接收通知的电话号码

\* @param notifyType 通知类别，通过该枚举值在配置文件中获取相应的模版ID

\* @param params 通知模版内容里的参数，类似"您的验证码为{1}"中{1}的值

\* @return

\*/

public SmsResult notifySmsTemplateSync(String phoneNumber, NotifyType notifyType, String[] params) {

if (smsSender == null)

return null;

int templateId = Integer.parseInt(getTemplateId(notifyType, smsTemplate));

return smsSender.sendWithTemplate(phoneNumber, templateId, params);

}

/\*\*

\* 微信模版消息通知,不跳转

\* <p>

\* 该方法会尝试从数据库获取缓存的FormId去发送消息

\*

\* @param touser 接收者openId

\* @param notifyType 通知类别，通过该枚举值在配置文件中获取相应的模版ID

\* @param params 通知模版内容里的参数，类似"您的验证码为{1}"中{1}的值

\*/

@Async

public void notifyWxTemplate(String touser, NotifyType notifyType, String[] params) {

if (wxTemplateSender == null)

return;

String templateId = getTemplateId(notifyType, wxTemplate);

wxTemplateSender.sendWechatMsg(touser, templateId, params);

}

/\*\*

\* 微信模版消息通知，带跳转

\* <p>

\* 该方法会尝试从数据库获取缓存的FormId去发送消息

\*

\* @param touser 接收者openId

\* @param notifyType 通知类别，通过该枚举值在配置文件中获取相应的模版ID

\* @param params 通知模版内容里的参数，类似"您的验证码为{1}"中{1}的值

\* @param page 点击消息跳转的页面

\*/

@Async

public void notifyWxTemplate(String touser, NotifyType notifyType, String[] params, String page) {

if (wxTemplateSender == null)

return;

String templateId = getTemplateId(notifyType, wxTemplate);

wxTemplateSender.sendWechatMsg(touser, templateId, params, page);

}

/\*\*

\* 邮件消息通知,

\* 接收者在spring.mail.sendto中指定

\*

\* @param subject 邮件标题

\* @param content 邮件内容

\*/

@Async

public void notifyMail(String subject, String content) {

if (mailSender == null)

return;

SimpleMailMessage message = new SimpleMailMessage();

message.setFrom(sendFrom);

message.setTo(sendTo);

message.setSubject(subject);

message.setText(content);

mailSender.send(message);

}

private String getTemplateId(NotifyType notifyType, List<Map<String, String>> values) {

for (Map<String, String> item : values) {

String notifyTypeStr = notifyType.getType();

if (item.get("name").equals(notifyTypeStr))

return item.get("templateId");

}

return null;

}

public void setMailSender(MailSender mailSender) {

this.mailSender = mailSender;

}

public void setSendFrom(String sendFrom) {

this.sendFrom = sendFrom;

}

public void setSendTo(String sendTo) {

this.sendTo = sendTo;

}

public void setSmsSender(SmsSender smsSender) {

this.smsSender = smsSender;

}

public void setSmsTemplate(List<Map<String, String>> smsTemplate) {

this.smsTemplate = smsTemplate;

}

public void setWxTemplateSender(WxTemplateSender wxTemplateSender) {

this.wxTemplateSender = wxTemplateSender;

}

public void setWxTemplate(List<Map<String, String>> wxTemplate) {

this.wxTemplate = wxTemplate;

}

}

package org.linlinjava.litemall.core.notify;

public enum NotifyType {

PAY\_SUCCEED("paySucceed"),

SHIP("ship"),

REFUND("refund"),

CAPTCHA("captcha");

private String type;

NotifyType(String type) {

this.type = type;

}

public String getType() {

return this.type;

}

}

package org.linlinjava.litemall.core.notify;

/\*\*

\* 发送短信的返回结果

\*/

public class SmsResult {

private boolean successful;

private Object result;

/\*\*

\* 短信是否发送成功

\*

\* @return

\*/

public boolean isSuccessful() {

return successful;

}

public void setSuccessful(boolean successful) {

this.successful = successful;

}

public Object getResult() {

return result;

}

public void setResult(Object result) {

this.result = result;

}

}

package org.linlinjava.litemall.core.notify;

public interface SmsSender {

/\*\*

\* 发送短信息

\*

\* @param phone 接收通知的电话号码

\* @param content 短消息内容

\*/

SmsResult send(String phone, String content);

/\*\*

\* 通过短信模版发送短信息

\*

\* @param phone 接收通知的电话号码

\* @param templateId 通知模板ID

\* @param params 通知模版内容里的参数，类似"您的验证码为{1}"中{1}的值

\*/

SmsResult sendWithTemplate(String phone, int templateId, String[] params);

}

package org.linlinjava.litemall.core.notify;

import com.github.qcloudsms.SmsSingleSender;

import com.github.qcloudsms.SmsSingleSenderResult;

import com.github.qcloudsms.httpclient.HTTPException;

import org.apache.commons.logging.Log;

import org.apache.commons.logging.LogFactory;

import java.io.IOException;

/\*

\* 腾讯云短信服务

\*/

public class TencentSmsSender implements SmsSender {

private final Log logger = LogFactory.getLog(TencentSmsSender.class);

private SmsSingleSender sender;

public SmsSingleSender getSender() {

return sender;

}

public void setSender(SmsSingleSender sender) {

this.sender = sender;

}

@Override

public SmsResult send(String phone, String content) {

try {

SmsSingleSenderResult result = sender.send(0, "86", phone, content, "", "");

logger.debug(result);

SmsResult smsResult = new SmsResult();

smsResult.setSuccessful(true);

smsResult.setResult(result);

return smsResult;

} catch (HTTPException | IOException e) {

e.printStackTrace();

}

return null;

}

@Override

public SmsResult sendWithTemplate(String phone, int templateId, String[] params) {

try {

SmsSingleSenderResult result = sender.sendWithParam("86", phone, templateId, params, "", "", "");

logger.debug(result);

SmsResult smsResult = new SmsResult();

smsResult.setSuccessful(true);

smsResult.setResult(result);

return smsResult;

} catch (HTTPException | IOException e) {

e.printStackTrace();

}

return null;

}

}

package org.linlinjava.litemall.core.notify;

import cn.binarywang.wx.miniapp.api.WxMaService;

import cn.binarywang.wx.miniapp.bean.WxMaTemplateData;

import cn.binarywang.wx.miniapp.bean.WxMaTemplateMessage;

import org.apache.commons.logging.Log;

import org.apache.commons.logging.LogFactory;

import org.linlinjava.litemall.db.domain.LitemallUserFormid;

import org.linlinjava.litemall.db.service.LitemallUserFormIdService;

import org.springframework.beans.factory.annotation.Autowired;

import java.util.ArrayList;

import java.util.List;

/\*\*

\* 微信模版消息通知

\*/

public class WxTemplateSender {

private final Log logger = LogFactory.getLog(WxTemplateSender.class);

@Autowired

private WxMaService wxMaService;

@Autowired

private LitemallUserFormIdService formIdService;

/\*\*

\* 发送微信消息(模板消息),不带跳转

\*

\* @param touser 用户 OpenID

\* @param templatId 模板消息ID

\* @param parms 详细内容

\*/

public void sendWechatMsg(String touser, String templatId, String[] parms) {

sendMsg(touser, templatId, parms, "", "", "");

}

/\*\*

\* 发送微信消息(模板消息),带跳转

\*

\* @param touser 用户 OpenID

\* @param templatId 模板消息ID

\* @param parms 详细内容

\* @param page 跳转页面

\*/

public void sendWechatMsg(String touser, String templatId, String[] parms, String page) {

sendMsg(touser, templatId, parms, page, "", "");

}

private void sendMsg(String touser, String templatId, String[] parms, String page, String color, String emphasisKeyword) {

LitemallUserFormid userFormid = formIdService.queryByOpenId(touser);

if (userFormid == null)

return;

WxMaTemplateMessage msg = new WxMaTemplateMessage();

msg.setTemplateId(templatId);

msg.setToUser(touser);

msg.setFormId(userFormid.getFormid());

msg.setPage(page);

msg.setColor(color);

msg.setEmphasisKeyword(emphasisKeyword);

msg.setData(createMsgData(parms));

try {

wxMaService.getMsgService().sendTemplateMsg(msg);

if (formIdService.updateUserFormId(userFormid) == 0) {

logger.warn("更新数据已失效");

}

} catch (Exception e) {

e.printStackTrace();

}

}

private List<WxMaTemplateData> createMsgData(String[] parms) {

List<WxMaTemplateData> dataList = new ArrayList<WxMaTemplateData>();

for (int i = 1; i <= parms.length; i++) {

dataList.add(new WxMaTemplateData("keyword" + i, parms[i - 1]));

}

return dataList;

}

}

package org.linlinjava.litemall.core.qcode;

import cn.binarywang.wx.miniapp.api.WxMaService;

import me.chanjar.weixin.common.error.WxErrorException;

import org.linlinjava.litemall.core.storage.StorageService;

import org.linlinjava.litemall.core.system.SystemConfig;

import org.linlinjava.litemall.db.domain.LitemallGroupon;

import org.linlinjava.litemall.db.domain.LitemallStorage;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.core.io.ClassPathResource;

import org.springframework.stereotype.Service;

import javax.imageio.ImageIO;

import java.awt.\*;

import java.awt.image.BufferedImage;

import java.io.\*;

import java.net.URL;

@Service

public class QCodeService {

@Autowired

WxMaService wxMaService;

@Autowired

private StorageService storageService;

public String createGrouponShareImage(String goodName, String goodPicUrl, LitemallGroupon groupon) {

try {

//创建该商品的二维码

File file = wxMaService.getQrcodeService().createWxaCodeUnlimit("groupon," + groupon.getId(), "pages/index/index");

FileInputStream inputStream = new FileInputStream(file);

//将商品图片，商品名字,商城名字画到模版图中

byte[] imageData = drawPicture(inputStream, goodPicUrl, goodName);

ByteArrayInputStream inputStream2 = new ByteArrayInputStream(imageData);

//存储分享图

LitemallStorage storageInfo = storageService.store(inputStream2, imageData.length, "image/jpeg", getKeyName(groupon.getId().toString()));

return storageInfo.getUrl();

} catch (WxErrorException e) {

e.printStackTrace();

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

return "";

}

/\*\*

\* 创建商品分享图

\*

\* @param goodId

\* @param goodPicUrl

\* @param goodName

\*/

public String createGoodShareImage(String goodId, String goodPicUrl, String goodName) {

if (!SystemConfig.isAutoCreateShareImage())

return "";

try {

//创建该商品的二维码

File file = wxMaService.getQrcodeService().createWxaCodeUnlimit("goods," + goodId, "pages/index/index");

FileInputStream inputStream = new FileInputStream(file);

//将商品图片，商品名字,商城名字画到模版图中

byte[] imageData = drawPicture(inputStream, goodPicUrl, goodName);

ByteArrayInputStream inputStream2 = new ByteArrayInputStream(imageData);

//存储分享图

LitemallStorage litemallStorage = storageService.store(inputStream2, imageData.length, "image/jpeg", getKeyName(goodId));

return litemallStorage.getUrl();

} catch (WxErrorException e) {

e.printStackTrace();

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

return "";

}

private String getKeyName(String goodId) {

return "GOOD\_QCODE\_" + goodId + ".jpg";

}

/\*\*

\* 将商品图片，商品名字画到模版图中

\*

\* @param qrCodeImg 二维码图片

\* @param goodPicUrl 商品图片地址

\* @param goodName 商品名称

\* @return

\* @throws IOException

\*/

private byte[] drawPicture(InputStream qrCodeImg, String goodPicUrl, String goodName) throws IOException {

//底图

ClassPathResource redResource = new ClassPathResource("back.png");

BufferedImage red = ImageIO.read(redResource.getInputStream());

//商品图片

URL goodPic = new URL(goodPicUrl);

BufferedImage goodImage = ImageIO.read(goodPic);

//小程序二维码

BufferedImage qrCodeImage = ImageIO.read(qrCodeImg);

// --- 画图 ---

//底层空白 bufferedImage

BufferedImage baseImage = new BufferedImage(red.getWidth(), red.getHeight(), BufferedImage.TYPE\_4BYTE\_ABGR\_PRE);

//画上图片

drawImgInImg(baseImage, red, 0, 0, red.getWidth(), red.getHeight());

//画上商品图片

drawImgInImg(baseImage, goodImage, 71, 69, 660, 660);

//画上小程序二维码

drawImgInImg(baseImage, qrCodeImage, 448, 767, 300, 300);

//写上商品名称

drawTextInImg(baseImage, goodName, 65, 867);

//写上商城名称

// drawTextInImgCenter(baseImage, shopName, 98);

//转jpg

BufferedImage result = new BufferedImage(baseImage.getWidth(), baseImage

.getHeight(), BufferedImage.TYPE\_3BYTE\_BGR);

result.getGraphics().drawImage(baseImage, 0, 0, null);

ByteArrayOutputStream bs = new ByteArrayOutputStream();

ImageIO.write(result, "jpg", bs);

//最终byte数组

return bs.toByteArray();

}

private void drawTextInImgCenter(BufferedImage baseImage, String textToWrite, int y) {

Graphics2D g2D = (Graphics2D) baseImage.getGraphics();

g2D.setColor(new Color(167, 136, 69));

String fontName = "Microsoft YaHei";

Font f = new Font(fontName, Font.PLAIN, 28);

g2D.setFont(f);

g2D.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);

// 计算文字长度，计算居中的x点坐标

FontMetrics fm = g2D.getFontMetrics(f);

int textWidth = fm.stringWidth(textToWrite);

int widthX = (baseImage.getWidth() - textWidth) / 2;

// 表示这段文字在图片上的位置(x,y) .第一个是你设置的内容。

g2D.drawString(textToWrite, widthX, y);

// 释放对象

g2D.dispose();

}

private void drawTextInImg(BufferedImage baseImage, String textToWrite, int x, int y) {

Graphics2D g2D = (Graphics2D) baseImage.getGraphics();

g2D.setColor(new Color(167, 136, 69));

//TODO 注意，这里的字体必须安装在服务器上

g2D.setFont(new Font("Microsoft YaHei", Font.PLAIN, 28));

g2D.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);

g2D.drawString(textToWrite, x, y);

g2D.dispose();

}

private void drawImgInImg(BufferedImage baseImage, BufferedImage imageToWrite, int x, int y, int width, int heigth) {

Graphics2D g2D = (Graphics2D) baseImage.getGraphics();

g2D.drawImage(imageToWrite, x, y, width, heigth, null);

g2D.dispose();

}

}

package org.linlinjava.litemall.core.storage.config;

import org.linlinjava.litemall.core.storage.\*;

import org.springframework.boot.context.properties.EnableConfigurationProperties;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

@Configuration

@EnableConfigurationProperties(StorageProperties.class)

public class StorageAutoConfiguration {

private final StorageProperties properties;

public StorageAutoConfiguration(StorageProperties properties) {

this.properties = properties;

}

@Bean

public StorageService storageService() {

StorageService storageService = new StorageService();

String active = this.properties.getActive();

storageService.setActive(active);

if (active.equals("local")) {

storageService.setStorage(localStorage());

} else if (active.equals("aliyun")) {

storageService.setStorage(aliyunStorage());

} else if (active.equals("tencent")) {

storageService.setStorage(tencentStorage());

} else if (active.equals("qiniu")) {

storageService.setStorage(qiniuStorage());

} else {

throw new RuntimeException("当前存储模式 " + active + " 不支持");

}

return storageService;

}

@Bean

public LocalStorage localStorage() {

LocalStorage localStorage = new LocalStorage();

StorageProperties.Local local = this.properties.getLocal();

localStorage.setAddress(local.getAddress());

localStorage.setStoragePath(local.getStoragePath());

return localStorage;

}

@Bean

public AliyunStorage aliyunStorage() {

AliyunStorage aliyunStorage = new AliyunStorage();

StorageProperties.Aliyun aliyun = this.properties.getAliyun();

aliyunStorage.setAccessKeyId(aliyun.getAccessKeyId());

aliyunStorage.setAccessKeySecret(aliyun.getAccessKeySecret());

aliyunStorage.setBucketName(aliyun.getBucketName());

aliyunStorage.setEndpoint(aliyun.getEndpoint());

return aliyunStorage;

}

@Bean

public TencentStorage tencentStorage() {

TencentStorage tencentStorage = new TencentStorage();

StorageProperties.Tencent tencent = this.properties.getTencent();

tencentStorage.setSecretId(tencent.getSecretId());

tencentStorage.setSecretKey(tencent.getSecretKey());

tencentStorage.setBucketName(tencent.getBucketName());

tencentStorage.setRegion(tencent.getRegion());

return tencentStorage;

}

@Bean

public QiniuStorage qiniuStorage() {

QiniuStorage qiniuStorage = new QiniuStorage();

StorageProperties.Qiniu qiniu = this.properties.getQiniu();

qiniuStorage.setAccessKey(qiniu.getAccessKey());

qiniuStorage.setSecretKey(qiniu.getSecretKey());

qiniuStorage.setBucketName(qiniu.getBucketName());

qiniuStorage.setEndpoint(qiniu.getEndpoint());

return qiniuStorage;

}

}

package org.linlinjava.litemall.core.storage.config;

import org.springframework.boot.context.properties.ConfigurationProperties;

@ConfigurationProperties(prefix = "litemall.storage")

public class StorageProperties {

private String active;

private Local local;

private Aliyun aliyun;

private Tencent tencent;

private Qiniu qiniu;

public String getActive() {

return active;

}

public void setActive(String active) {

this.active = active;

}

public Local getLocal() {

return local;

}

public void setLocal(Local local) {

this.local = local;

}

public Aliyun getAliyun() {

return aliyun;

}

public void setAliyun(Aliyun aliyun) {

this.aliyun = aliyun;

}

public Tencent getTencent() {

return tencent;

}

public void setTencent(Tencent tencent) {

this.tencent = tencent;

}

public Qiniu getQiniu() {

return qiniu;

}

public void setQiniu(Qiniu qiniu) {

this.qiniu = qiniu;

}

public static class Local {

private String address;

private String storagePath;

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String getStoragePath() {

return storagePath;

}

public void setStoragePath(String storagePath) {

this.storagePath = storagePath;

}

}

public static class Tencent {

private String secretId;

private String secretKey;

private String region;

private String bucketName;

public String getSecretId() {

return secretId;

}

public void setSecretId(String secretId) {

this.secretId = secretId;

}

public String getSecretKey() {

return secretKey;

}

public void setSecretKey(String secretKey) {

this.secretKey = secretKey;

}

public String getRegion() {

return region;

}

public void setRegion(String region) {

this.region = region;

}

public String getBucketName() {

return bucketName;

}

public void setBucketName(String bucketName) {

this.bucketName = bucketName;

}

}

public static class Aliyun {

private String endpoint;

private String accessKeyId;

private String accessKeySecret;

private String bucketName;

public String getEndpoint() {

return endpoint;

}

public void setEndpoint(String endpoint) {

this.endpoint = endpoint;

}

public String getAccessKeyId() {

return accessKeyId;

}

public void setAccessKeyId(String accessKeyId) {

this.accessKeyId = accessKeyId;

}

public String getAccessKeySecret() {

return accessKeySecret;

}

public void setAccessKeySecret(String accessKeySecret) {

this.accessKeySecret = accessKeySecret;

}

public String getBucketName() {

return bucketName;

}

public void setBucketName(String bucketName) {

this.bucketName = bucketName;

}

}

public static class Qiniu {

private String endpoint;

private String accessKey;

private String secretKey;

private String bucketName;

public String getEndpoint() {

return endpoint;

}

public void setEndpoint(String endpoint) {

this.endpoint = endpoint;

}

public String getAccessKey() {

return accessKey;

}

public void setAccessKey(String accessKey) {

this.accessKey = accessKey;

}

public String getSecretKey() {

return secretKey;

}

public void setSecretKey(String secretKey) {

this.secretKey = secretKey;

}

public String getBucketName() {

return bucketName;

}

public void setBucketName(String bucketName) {

this.bucketName = bucketName;

}

}

}

package org.linlinjava.litemall.core.storage;

import com.aliyun.oss.OSSClient;

import com.aliyun.oss.model.ObjectMetadata;

import com.aliyun.oss.model.PutObjectRequest;

import com.aliyun.oss.model.PutObjectResult;

import org.springframework.core.io.Resource;

import org.springframework.core.io.UrlResource;

import java.io.InputStream;

import java.net.MalformedURLException;

import java.net.URL;

import java.nio.file.Path;

import java.util.stream.Stream;

/\*\*

\* @author Yogeek

\* @date 2018/7/16 16:10

\* @decrpt 阿里云对象存储服务

\*/

public class AliyunStorage implements Storage {

private String endpoint;

private String accessKeyId;

private String accessKeySecret;

private String bucketName;

public String getEndpoint() {

return endpoint;

}

public void setEndpoint(String endpoint) {

this.endpoint = endpoint;

}

public String getAccessKeyId() {

return accessKeyId;

}

public void setAccessKeyId(String accessKeyId) {

this.accessKeyId = accessKeyId;

}

public String getAccessKeySecret() {

return accessKeySecret;

}

public void setAccessKeySecret(String accessKeySecret) {

this.accessKeySecret = accessKeySecret;

}

public String getBucketName() {

return bucketName;

}

public void setBucketName(String bucketName) {

this.bucketName = bucketName;

}

/\*\*

\* 获取阿里云OSS客户端对象

\*

\* @return ossClient

\*/

private OSSClient getOSSClient() {

return new OSSClient(endpoint, accessKeyId, accessKeySecret);

}

private String getBaseUrl() {

return "https://" + bucketName + "." + endpoint + "/";

}

/\*\*

\* 阿里云OSS对象存储简单上传实现

\*/

@Override

public void store(InputStream inputStream, long contentLength, String contentType, String keyName) {

try {

// 简单文件上传, 最大支持 5 GB, 适用于小文件上传, 建议 20M以下的文件使用该接口

ObjectMetadata objectMetadata = new ObjectMetadata();

objectMetadata.setContentLength(contentLength);

objectMetadata.setContentType(contentType);

// 对象键（Key）是对象在存储桶中的唯一标识。

PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, keyName, inputStream, objectMetadata);

PutObjectResult putObjectResult = getOSSClient().putObject(putObjectRequest);

} catch (Exception ex) {

ex.printStackTrace();

}

}

@Override

public Stream<Path> loadAll() {

return null;

}

@Override

public Path load(String keyName) {

return null;

}

@Override

public Resource loadAsResource(String keyName) {

try {

URL url = new URL(getBaseUrl() + keyName);

Resource resource = new UrlResource(url);

if (resource.exists() || resource.isReadable()) {

return resource;

} else {

return null;

}

} catch (MalformedURLException e) {

e.printStackTrace();

return null;

}

}

@Override

public void delete(String keyName) {

try {

getOSSClient().deleteObject(bucketName, keyName);

} catch (Exception e) {

e.printStackTrace();

}

}

@Override

public String generateUrl(String keyName) {

return getBaseUrl() + keyName;

}

}

package org.linlinjava.litemall.core.storage;

import org.springframework.core.io.Resource;

import org.springframework.core.io.UrlResource;

import java.io.IOException;

import java.io.InputStream;

import java.net.MalformedURLException;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.nio.file.StandardCopyOption;

import java.util.stream.Stream;

/\*\*

\* 服务器本地对象存储服务

\*/

public class LocalStorage implements Storage {

private String storagePath;

private String address;

private Path rootLocation;

public String getStoragePath() {

return storagePath;

}

public void setStoragePath(String storagePath) {

this.storagePath = storagePath;

this.rootLocation = Paths.get(storagePath);

try {

Files.createDirectories(rootLocation);

} catch (IOException e) {

e.printStackTrace();

}

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

@Override

public void store(InputStream inputStream, long contentLength, String contentType, String keyName) {

try {

Files.copy(inputStream, rootLocation.resolve(keyName), StandardCopyOption.REPLACE\_EXISTING);

} catch (IOException e) {

throw new RuntimeException("Failed to store file " + keyName, e);

}

}

@Override

public Stream<Path> loadAll() {

try {

return Files.walk(rootLocation, 1)

.filter(path -> !path.equals(rootLocation))

.map(path -> rootLocation.relativize(path));

} catch (IOException e) {

throw new RuntimeException("Failed to read stored files", e);

}

}

@Override

public Path load(String filename) {

return rootLocation.resolve(filename);

}

@Override

public Resource loadAsResource(String filename) {

try {

Path file = load(filename);

Resource resource = new UrlResource(file.toUri());

if (resource.exists() || resource.isReadable()) {

return resource;

} else {

return null;

}

} catch (MalformedURLException e) {

e.printStackTrace();

return null;

}

}

@Override

public void delete(String filename) {

Path file = load(filename);

try {

Files.delete(file);

} catch (IOException e) {

e.printStackTrace();

}

}

@Override

public String generateUrl(String keyName) {

String url = address + keyName;

return url;

}

}

package org.linlinjava.litemall.core.storage;

import com.qiniu.common.QiniuException;

import com.qiniu.http.Response;

import com.qiniu.storage.BucketManager;

import com.qiniu.storage.Configuration;

import com.qiniu.storage.UploadManager;

import com.qiniu.util.Auth;

import org.springframework.core.io.Resource;

import org.springframework.core.io.UrlResource;

import java.io.InputStream;

import java.net.MalformedURLException;

import java.net.URL;

import java.nio.file.Path;

import java.util.stream.Stream;

public class QiniuStorage implements Storage {

private String endpoint;

private String accessKey;

private String secretKey;

private String bucketName;

private Auth auth;

private UploadManager uploadManager;

private BucketManager bucketManager;

public String getEndpoint() {

return endpoint;

}

public void setEndpoint(String endpoint) {

this.endpoint = endpoint;

}

public String getAccessKey() {

return accessKey;

}

public void setAccessKey(String accessKey) {

this.accessKey = accessKey;

}

public String getSecretKey() {

return secretKey;

}

public void setSecretKey(String secretKey) {

this.secretKey = secretKey;

}

public String getBucketName() {

return bucketName;

}

public void setBucketName(String bucketName) {

this.bucketName = bucketName;

}

/\*\*

\* 七牛云OSS对象存储简单上传实现

\*/

@Override

public void store(InputStream inputStream, long contentLength, String contentType, String keyName) {

if (uploadManager == null) {

if (auth == null) {

auth = Auth.create(accessKey, secretKey);

}

// uploadManager = new UploadManager(new Configuration());

}

try {

String upToken = auth.uploadToken(bucketName);

Response response = uploadManager.put(inputStream, keyName, upToken, null, contentType);

} catch (QiniuException ex) {

ex.printStackTrace();

}

}

@Override

public Stream<Path> loadAll() {

return null;

}

@Override

public Path load(String keyName) {

return null;

}

@Override

public Resource loadAsResource(String keyName) {

try {

URL url = new URL(generateUrl(keyName));

Resource resource = new UrlResource(url);

if (resource.exists() || resource.isReadable()) {

return resource;

} else {

return null;

}

} catch (MalformedURLException e) {

e.printStackTrace();

return null;

}

}

@Override

public void delete(String keyName) {

if (bucketManager == null) {

if (auth == null) {

auth = Auth.create(accessKey, secretKey);

}

// bucketManager = new BucketManager(auth, new Configuration());

}

try {

bucketManager.delete(bucketName, keyName);

} catch (Exception e) {

e.printStackTrace();

}

}

@Override

public String generateUrl(String keyName) {

return endpoint + "/" + keyName;

}

}

package org.linlinjava.litemall.core.storage;

import org.springframework.core.io.Resource;

import java.io.InputStream;

import java.nio.file.Path;

import java.util.stream.Stream;

/\*\*

\* 对象存储接口

\*/

public interface Storage {

/\*\*

\* 存储一个文件对象

\*

\* @param inputStream 文件输入流

\* @param contentLength 文件长度

\* @param contentType 文件类型

\* @param keyName 文件名

\*/

void store(InputStream inputStream, long contentLength, String contentType, String keyName);

Stream<Path> loadAll();

Path load(String keyName);

Resource loadAsResource(String keyName);

void delete(String keyName);

String generateUrl(String keyName);

}

package org.linlinjava.litemall.core.storage;

import org.linlinjava.litemall.core.util.CharUtil;

import org.linlinjava.litemall.db.domain.LitemallStorage;

import org.linlinjava.litemall.db.service.LitemallStorageService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.core.io.Resource;

import java.io.InputStream;

import java.nio.file.Path;

import java.util.stream.Stream;

/\*\*

\* 提供存储服务类，所有存储服务均由该类对外提供

\*/

public class StorageService {

private String active;

private Storage storage;

@Autowired

private LitemallStorageService litemallStorageService;

public String getActive() {

return active;

}

public void setActive(String active) {

this.active = active;

}

public Storage getStorage() {

return storage;

}

public void setStorage(Storage storage) {

this.storage = storage;

}

/\*\*

\* 存储一个文件对象

\*

\* @param inputStream 文件输入流

\* @param contentLength 文件长度

\* @param contentType 文件类型

\* @param fileName 文件索引名

\*/

public LitemallStorage store(InputStream inputStream, long contentLength, String contentType, String fileName) {

String key = generateKey(fileName);

storage.store(inputStream, contentLength, contentType, key);

String url = generateUrl(key);

LitemallStorage storageInfo = new LitemallStorage();

storageInfo.setName(fileName);

storageInfo.setSize((int) contentLength);

storageInfo.setType(contentType);

storageInfo.setKey(key);

storageInfo.setUrl(url);

litemallStorageService.add(storageInfo);

return storageInfo;

}

private String generateKey(String originalFilename) {

int index = originalFilename.lastIndexOf('.');

String suffix = originalFilename.substring(index);

String key = null;

LitemallStorage storageInfo = null;

do {

key = CharUtil.getRandomString(20) + suffix;

storageInfo = litemallStorageService.findByKey(key);

}

while (storageInfo != null);

return key;

}

public Stream<Path> loadAll() {

return storage.loadAll();

}

public Path load(String keyName) {

return storage.load(keyName);

}

public Resource loadAsResource(String keyName) {

return storage.loadAsResource(keyName);

}

public void delete(String keyName) {

storage.delete(keyName);

}

private String generateUrl(String keyName) {

return storage.generateUrl(keyName);

}

}

package org.linlinjava.litemall.core.storage;

import com.qcloud.cos.COSClient;

import com.qcloud.cos.ClientConfig;

import com.qcloud.cos.auth.BasicCOSCredentials;

import com.qcloud.cos.auth.COSCredentials;

import com.qcloud.cos.model.ObjectMetadata;

import com.qcloud.cos.model.PutObjectRequest;

import com.qcloud.cos.model.PutObjectResult;

import com.qcloud.cos.region.Region;

import org.springframework.core.io.Resource;

import org.springframework.core.io.UrlResource;

import java.io.InputStream;

import java.net.MalformedURLException;

import java.net.URL;

import java.nio.file.Path;

import java.util.stream.Stream;

/\*\*

\* 腾讯对象存储服务

\*/

public class TencentStorage implements Storage {

private String secretId;

private String secretKey;

private String region;

private String bucketName;

private COSClient cosClient;

public String getSecretId() {

return secretId;

}

public void setSecretId(String secretId) {

this.secretId = secretId;

}

public String getSecretKey() {

return secretKey;

}

public void setSecretKey(String secretKey) {

this.secretKey = secretKey;

}

public String getRegion() {

return region;

}

public void setRegion(String region) {

this.region = region;

}

public String getBucketName() {

return bucketName;

}

public void setBucketName(String bucketName) {

this.bucketName = bucketName;

}

private COSClient getCOSClient() {

if (cosClient == null) {

// 1 初始化用户身份信息(secretId, secretKey)

COSCredentials cred = new BasicCOSCredentials(secretId, secretKey);

// 2 设置bucket的区域, COS地域的简称请参照 https://cloud.tencent.com/document/product/436/6224

ClientConfig clientConfig = new ClientConfig(new Region(region));

cosClient = new COSClient(cred, clientConfig);

}

return cosClient;

}

private String getBaseUrl() {

return "https://" + bucketName + ".cos-website." + region + ".myqcloud.com/";

}

@Override

public void store(InputStream inputStream, long contentLength, String contentType, String keyName) {

try {

// 简单文件上传, 最大支持 5 GB, 适用于小文件上传, 建议 20M以下的文件使用该接口

ObjectMetadata objectMetadata = new ObjectMetadata();

objectMetadata.setContentLength(contentLength);

objectMetadata.setContentType(contentType);

// 对象键（Key）是对象在存储桶中的唯一标识。例如，在对象的访问域名 `bucket1-1250000000.cos.ap-guangzhou.myqcloud.com/doc1/pic1.jpg` 中，对象键为 doc1/pic1.jpg, 详情参考 [对象键](https://cloud.tencent.com/document/product/436/13324)

PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, keyName, inputStream, objectMetadata);

PutObjectResult putObjectResult = getCOSClient().putObject(putObjectRequest);

} catch (Exception ex) {

ex.printStackTrace();

}

}

@Override

public Stream<Path> loadAll() {

return null;

}

@Override

public Path load(String keyName) {

return null;

}

@Override

public Resource loadAsResource(String keyName) {

try {

URL url = new URL(getBaseUrl() + keyName);

Resource resource = new UrlResource(url);

if (resource.exists() || resource.isReadable()) {

return resource;

} else {

return null;

}

} catch (MalformedURLException e) {

e.printStackTrace();

return null;

}

}

@Override

public void delete(String keyName) {

try {

getCOSClient().deleteObject(bucketName, keyName);

} catch (Exception e) {

e.printStackTrace();

}

}

@Override

public String generateUrl(String keyName) {

return getBaseUrl() + keyName;

}

}

package org.linlinjava.litemall.core.system;

import java.math.BigDecimal;

import java.util.HashMap;

import java.util.Map;

/\*\*

\* 系统设置

\*/

public class SystemConfig {

// 小程序相关配置

public final static String LITEMALL\_WX\_INDEX\_NEW = "litemall\_wx\_index\_new";

public final static String LITEMALL\_WX\_INDEX\_HOT = "litemall\_wx\_index\_hot";

public final static String LITEMALL\_WX\_INDEX\_BRAND = "litemall\_wx\_index\_brand";

public final static String LITEMALL\_WX\_INDEX\_TOPIC = "litemall\_wx\_index\_topic";

public final static String LITEMALL\_WX\_INDEX\_CATLOG\_LIST = "litemall\_wx\_catlog\_list";

public final static String LITEMALL\_WX\_INDEX\_CATLOG\_GOODS = "litemall\_wx\_catlog\_goods";

public final static String LITEMALL\_WX\_SHARE = "litemall\_wx\_share";

// 运费相关配置

public final static String LITEMALL\_EXPRESS\_FREIGHT\_VALUE = "litemall\_express\_freight\_value";

public final static String LITEMALL\_EXPRESS\_FREIGHT\_MIN = "litemall\_express\_freight\_min";

// 订单相关配置

public final static String LITEMALL\_ORDER\_UNPAID = "litemall\_order\_unpaid";

public final static String LITEMALL\_ORDER\_UNCONFIRM = "litemall\_order\_unconfirm";

public final static String LITEMALL\_ORDER\_COMMENT = "litemall\_order\_comment";

// 商场相关配置

public final static String LITEMALL\_MALL\_NAME = "litemall\_mall\_name";

public final static String LITEMALL\_MALL\_ADDRESS = "litemall\_mall\_address";

public final static String LITEMALL\_MALL\_PHONE = "litemall\_mall\_phone";

public final static String LITEMALL\_MALL\_QQ = "litemall\_mall\_qq";

//所有的配置均保存在该 HashMap 中

private static Map<String, String> SYSTEM\_CONFIGS = new HashMap<>();

private static String getConfig(String keyName) {

return SYSTEM\_CONFIGS.get(keyName);

}

private static Integer getConfigInt(String keyName) {

return Integer.parseInt(SYSTEM\_CONFIGS.get(keyName));

}

private static Boolean getConfigBoolean(String keyName) {

return Boolean.valueOf(SYSTEM\_CONFIGS.get(keyName));

}

private static BigDecimal getConfigBigDec(String keyName) {

return new BigDecimal(SYSTEM\_CONFIGS.get(keyName));

}

public static Integer getNewLimit() {

return getConfigInt(LITEMALL\_WX\_INDEX\_NEW);

}

public static Integer getHotLimit() {

return getConfigInt(LITEMALL\_WX\_INDEX\_HOT);

}

public static Integer getBrandLimit() {

return getConfigInt(LITEMALL\_WX\_INDEX\_BRAND);

}

public static Integer getTopicLimit() {

return getConfigInt(LITEMALL\_WX\_INDEX\_TOPIC);

}

public static Integer getCatlogListLimit() {

return getConfigInt(LITEMALL\_WX\_INDEX\_CATLOG\_LIST);

}

public static Integer getCatlogMoreLimit() {

return getConfigInt(LITEMALL\_WX\_INDEX\_CATLOG\_GOODS);

}

public static boolean isAutoCreateShareImage() {

return getConfigBoolean(LITEMALL\_WX\_SHARE);

}

public static BigDecimal getFreight() {

return getConfigBigDec(LITEMALL\_EXPRESS\_FREIGHT\_VALUE);

}

public static BigDecimal getFreightLimit() {

return getConfigBigDec(LITEMALL\_EXPRESS\_FREIGHT\_MIN);

}

public static Integer getOrderUnpaid() {

return getConfigInt(LITEMALL\_ORDER\_UNPAID);

}

public static Integer getOrderUnconfirm() {

return getConfigInt(LITEMALL\_ORDER\_UNCONFIRM);

}

public static Integer getOrderComment() {

return getConfigInt(LITEMALL\_ORDER\_COMMENT);

}

public static String getMallName() {

return getConfig(LITEMALL\_MALL\_NAME);

}

public static String getMallAddress() {

return getConfig(LITEMALL\_MALL\_ADDRESS);

}

public static String getMallPhone() {

return getConfig(LITEMALL\_MALL\_PHONE);

}

public static String getMallQQ() {

return getConfig(LITEMALL\_MALL\_QQ);

}

public static void setConfigs(Map<String, String> configs) {

SYSTEM\_CONFIGS = configs;

}

public static void updateConfigs(Map<String, String> data) {

for (Map.Entry<String, String> entry : data.entrySet()) {

SYSTEM\_CONFIGS.put(entry.getKey(), entry.getValue());

}

}

}

package org.linlinjava.litemall.core.system;

import org.linlinjava.litemall.core.util.SystemInfoPrinter;

import org.linlinjava.litemall.db.service.LitemallSystemConfigService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.core.env.Environment;

import org.springframework.stereotype.Component;

import javax.annotation.PostConstruct;

import java.util.HashMap;

import java.util.LinkedHashMap;

import java.util.Map;

/\*\*

\* 系统启动服务，用于设置系统配置信息、检查系统状态及打印系统信息

\*/

@Component

class SystemInistService {

@Autowired

private SystemInistService systemInistService;

@Autowired

private Environment environment;

@PostConstruct

private void inist() {

systemInistService = this;

initConfigs();

SystemInfoPrinter.printInfo("Litemall 初始化信息", getSystemInfo());

}

private final static Map<String, String> DEFAULT\_CONFIGS = new HashMap<>();

static {

// 小程序相关配置默认值

DEFAULT\_CONFIGS.put(SystemConfig.LITEMALL\_WX\_INDEX\_NEW, "6");

DEFAULT\_CONFIGS.put(SystemConfig.LITEMALL\_WX\_INDEX\_HOT, "6");

DEFAULT\_CONFIGS.put(SystemConfig.LITEMALL\_WX\_INDEX\_BRAND, "4");

DEFAULT\_CONFIGS.put(SystemConfig.LITEMALL\_WX\_INDEX\_TOPIC, "4");

DEFAULT\_CONFIGS.put(SystemConfig.LITEMALL\_WX\_INDEX\_CATLOG\_LIST, "4");

DEFAULT\_CONFIGS.put(SystemConfig.LITEMALL\_WX\_INDEX\_CATLOG\_GOODS, "4");

DEFAULT\_CONFIGS.put(SystemConfig.LITEMALL\_WX\_SHARE, "false");

// 运费相关配置默认值

DEFAULT\_CONFIGS.put(SystemConfig.LITEMALL\_EXPRESS\_FREIGHT\_VALUE, "8");

DEFAULT\_CONFIGS.put(SystemConfig.LITEMALL\_EXPRESS\_FREIGHT\_MIN, "88");

// 订单相关配置默认值

DEFAULT\_CONFIGS.put(SystemConfig.LITEMALL\_ORDER\_UNPAID, "30");

DEFAULT\_CONFIGS.put(SystemConfig.LITEMALL\_ORDER\_UNCONFIRM, "7");

DEFAULT\_CONFIGS.put(SystemConfig.LITEMALL\_ORDER\_COMMENT, "7");

// 订单相关配置默认值

DEFAULT\_CONFIGS.put(SystemConfig.LITEMALL\_MALL\_NAME, "litemall");

DEFAULT\_CONFIGS.put(SystemConfig.LITEMALL\_MALL\_ADDRESS, "上海");

DEFAULT\_CONFIGS.put(SystemConfig.LITEMALL\_MALL\_PHONE, "021-xxxx-xxxx");

DEFAULT\_CONFIGS.put(SystemConfig.LITEMALL\_MALL\_QQ, "738696120");

}

@Autowired

private LitemallSystemConfigService litemallSystemConfigService;

private void initConfigs() {

// 1. 读取数据库全部配置信息

Map<String, String> configs = litemallSystemConfigService.queryAll();

// 2. 分析DEFAULT\_CONFIGS

for (Map.Entry<String, String> entry : DEFAULT\_CONFIGS.entrySet()) {

if(configs.containsKey(entry.getKey())){

continue;

}

configs.put(entry.getKey(), entry.getValue());

litemallSystemConfigService.addConfig(entry.getKey(), entry.getValue());

}

SystemConfig.setConfigs(configs);

}

private Map<String, String> getSystemInfo() {

Map<String, String> infos = new LinkedHashMap<>();

infos.put(SystemInfoPrinter.CREATE\_PART\_COPPER + 0, "系统信息");

// 测试获取application-db.yml配置信息

infos.put("服务器端口", environment.getProperty("server.port"));

infos.put("数据库USER", environment.getProperty("spring.datasource.druid.username"));

infos.put("数据库地址", environment.getProperty("spring.datasource.druid.url"));

infos.put("调试级别", environment.getProperty("logging.level.org.linlinjava.litemall.wx"));

// 测试获取application-core.yml配置信息

infos.put(SystemInfoPrinter.CREATE\_PART\_COPPER + 1, "模块状态");

infos.put("邮件", environment.getProperty("litemall.notify.mail.enable"));

infos.put("短信", environment.getProperty("litemall.notify.sms.enable"));

infos.put("模版消息", environment.getProperty("litemall.notify.wx.enable"));

infos.put("快递信息", environment.getProperty("litemall.express.enable"));

infos.put("快递鸟ID", environment.getProperty("litemall.express.appId"));

infos.put("对象存储", environment.getProperty("litemall.storage.active"));

infos.put("本地对象存储路径", environment.getProperty("litemall.storage.local.storagePath"));

infos.put("本地对象访问地址", environment.getProperty("litemall.storage.local.address"));

infos.put("本地对象访问端口", environment.getProperty("litemall.storage.local.port"));

// 微信相关信息

infos.put(SystemInfoPrinter.CREATE\_PART\_COPPER + 2, "微信相关");

infos.put("微信APP KEY", environment.getProperty("litemall.wx.app-id"));

infos.put("微信APP-SECRET", environment.getProperty("litemall.wx.app-secret"));

infos.put("微信支付MCH-ID", environment.getProperty("litemall.wx.mch-id"));

infos.put("微信支付MCH-KEY", environment.getProperty("litemall.wx.mch-key"));

infos.put("微信支付通知地址", environment.getProperty("litemall.wx.notify-url"));

//测试获取System表配置信息

infos.put(SystemInfoPrinter.CREATE\_PART\_COPPER + 3, "系统设置");

infos.put("自动创建朋友圈分享图", Boolean.toString(SystemConfig.isAutoCreateShareImage()));

infos.put("商场名称", SystemConfig.getMallName());

infos.put("首页显示记录数：NEW,HOT,BRAND,TOPIC,CatlogList,CatlogMore", SystemConfig.getNewLimit() + "," + SystemConfig.getHotLimit() + "," + SystemConfig.getBrandLimit() + "," + SystemConfig.getTopicLimit() + "," + SystemConfig.getCatlogListLimit() + "," + SystemConfig.getCatlogMoreLimit());

return infos;

}

}

package org.linlinjava.litemall.core.util.bcrypt;  
// Copyright (c) 2006 Damien Miller <djm@mindrot.org>  
//  
// Permission to use, copy, modify, and distribute this software for any  
// purpose with or without fee is hereby granted, provided that the above  
// copyright notice and this permission notice appear in all copies.  
//  
// THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES  
// WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF  
// MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR  
// ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES  
// WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN  
// ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF  
// OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.  
  
import java.io.ByteArrayOutputStream;  
import java.io.UnsupportedEncodingException;  
import java.security.SecureRandom;  
  
/\*\*  
 \* BCrypt implements OpenBSD-style Blowfish password hashing using the scheme described in  
 \* "A Future-Adaptable Password Scheme" by Niels Provos and David Mazieres.  
 \* <p>  
 \* This password hashing system tries to thwart off-line password cracking using a  
 \* computationally-intensive hashing algorithm, based on Bruce Schneier's Blowfish cipher.  
 \* The work factor of the algorithm is parameterised, so it can be increased as computers  
 \* get faster.  
 \* <p>  
 \* Usage is really simple. To hash a password for the first time, call the hashpw method  
 \* with a random salt, like this:  
 \* <p>  
 \* <code>  
 \* String pw\_hash = BCrypt.hashpw(plain\_password, BCrypt.gensalt()); <br>  
 \* </code>  
 \* <p>  
 \* To check whether a plaintext password matches one that has been hashed previously, use  
 \* the checkpw method:  
 \* <p>  
 \* <code>  
 \* if (BCrypt.checkpw(candidate\_password, stored\_hash))<br>  
 \* &nbsp;&nbsp;&nbsp;&nbsp;System.out.println("It matches");<br>  
 \* else<br>  
 \* &nbsp;&nbsp;&nbsp;&nbsp;System.out.println("It does not match");<br>  
 \* </code>  
 \* <p>  
 \* The gensalt() method takes an optional parameter (log\_rounds) that determines the  
 \* computational complexity of the hashing:  
 \* <p>  
 \* <code>  
 \* String strong\_salt = BCrypt.gensalt(10)<br>  
 \* String stronger\_salt = BCrypt.gensalt(12)<br>  
 \* </code>  
 \* <p>  
 \* The amount of work increases exponentially (2\*\*log\_rounds), so each increment is twice  
 \* as much work. The default log\_rounds is 10, and the valid range is 4 to 31.  
 \*  
 \* @author Damien Miller  
 \*/  
public class BCrypt {  
    // BCrypt parameters  
  
    static final int MIN\_LOG\_ROUNDS = 4;  
    static final int MAX\_LOG\_ROUNDS = 31;  
    private static final int GENSALT\_DEFAULT\_LOG2\_ROUNDS = 10;  
    private static final int BCRYPT\_SALT\_LEN = 16;  
    // Blowfish parameters  
    private static final int BLOWFISH\_NUM\_ROUNDS = 16;  
    // Initial contents of key schedule  
    private static final int P\_orig[] = {0x243f6a88, 0x85a308d3, 0x13198a2e, 0x03707344,  
            0xa4093822, 0x299f31d0, 0x082efa98, 0xec4e6c89, 0x452821e6, 0x38d01377,  
            0xbe5466cf, 0x34e90c6c, 0xc0ac29b7, 0xc97c50dd, 0x3f84d5b5, 0xb5470917,  
            0x9216d5d9, 0x8979fb1b};  
    private static final int S\_orig[] = {0xd1310ba6, 0x98dfb5ac, 0x2ffd72db, 0xd01adfb7,  
            0xb8e1afed, 0x6a267e96, 0xba7c9045, 0xf12c7f99, 0x24a19947, 0xb3916cf7,  
            0x0801f2e2, 0x858efc16, 0x636920d8, 0x71574e69, 0xa458fea3, 0xf4933d7e,  
            0x0d95748f, 0x728eb658, 0x718bcd58, 0x82154aee, 0x7b54a41d, 0xc25a59b5,  
            0x9c30d539, 0x2af26013, 0xc5d1b023, 0x286085f0, 0xca417918, 0xb8db38ef,  
            0x8e79dcb0, 0x603a180e, 0x6c9e0e8b, 0xb01e8a3e, 0xd71577c1, 0xbd314b27,  
            0x78af2fda, 0x55605c60, 0xe65525f3, 0xaa55ab94, 0x57489862, 0x63e81440,  
            0x55ca396a, 0x2aab10b6, 0xb4cc5c34, 0x1141e8ce, 0xa15486af, 0x7c72e993,  
            0xb3ee1411, 0x636fbc2a, 0x2ba9c55d, 0x741831f6, 0xce5c3e16, 0x9b87931e,  
            0xafd6ba33, 0x6c24cf5c, 0x7a325381, 0x28958677, 0x3b8f4898, 0x6b4bb9af,  
            0xc4bfe81b, 0x66282193, 0x61d809cc, 0xfb21a991, 0x487cac60, 0x5dec8032,  
            0xef845d5d, 0xe98575b1, 0xdc262302, 0xeb651b88, 0x23893e81, 0xd396acc5,  
            0x0f6d6ff3, 0x83f44239, 0x2e0b4482, 0xa4842004, 0x69c8f04a, 0x9e1f9b5e,  
            0x21c66842, 0xf6e96c9a, 0x670c9c61, 0xabd388f0, 0x6a51a0d2, 0xd8542f68,  
            0x960fa728, 0xab5133a3, 0x6eef0b6c, 0x137a3be4, 0xba3bf050, 0x7efb2a98,  
            0xa1f1651d, 0x39af0176, 0x66ca593e, 0x82430e88, 0x8cee8619, 0x456f9fb4,  
            0x7d84a5c3, 0x3b8b5ebe, 0xe06f75d8, 0x85c12073, 0x401a449f, 0x56c16aa6,  
            0x4ed3aa62, 0x363f7706, 0x1bfedf72, 0x429b023d, 0x37d0d724, 0xd00a1248,  
            0xdb0fead3, 0x49f1c09b, 0x075372c9, 0x80991b7b, 0x25d479d8, 0xf6e8def7,  
            0xe3fe501a, 0xb6794c3b, 0x976ce0bd, 0x04c006ba, 0xc1a94fb6, 0x409f60c4,  
            0x5e5c9ec2, 0x196a2463, 0x68fb6faf, 0x3e6c53b5, 0x1339b2eb, 0x3b52ec6f,  
            0x6dfc511f, 0x9b30952c, 0xcc814544, 0xaf5ebd09, 0xbee3d004, 0xde334afd,  
            0x660f2807, 0x192e4bb3, 0xc0cba857, 0x45c8740f, 0xd20b5f39, 0xb9d3fbdb,  
            0x5579c0bd, 0x1a60320a, 0xd6a100c6, 0x402c7279, 0x679f25fe, 0xfb1fa3cc,  
            0x8ea5e9f8, 0xdb3222f8, 0x3c7516df, 0xfd616b15, 0x2f501ec8, 0xad0552ab,  
            0x323db5fa, 0xfd238760, 0x53317b48, 0x3e00df82, 0x9e5c57bb, 0xca6f8ca0,  
            0x1a87562e, 0xdf1769db, 0xd542a8f6, 0x287effc3, 0xac6732c6, 0x8c4f5573,  
            0x695b27b0, 0xbbca58c8, 0xe1ffa35d, 0xb8f011a0, 0x10fa3d98, 0xfd2183b8,  
            0x4afcb56c, 0x2dd1d35b, 0x9a53e479, 0xb6f84565, 0xd28e49bc, 0x4bfb9790,  
            0xe1ddf2da, 0xa4cb7e33, 0x62fb1341, 0xcee4c6e8, 0xef20cada, 0x36774c01,  
            0xd07e9efe, 0x2bf11fb4, 0x95dbda4d, 0xae909198, 0xeaad8e71, 0x6b93d5a0,  
            0xd08ed1d0, 0xafc725e0, 0x8e3c5b2f, 0x8e7594b7, 0x8ff6e2fb, 0xf2122b64,  
            0x8888b812, 0x900df01c, 0x4fad5ea0, 0x688fc31c, 0xd1cff191, 0xb3a8c1ad,  
            0x2f2f2218, 0xbe0e1777, 0xea752dfe, 0x8b021fa1, 0xe5a0cc0f, 0xb56f74e8,  
            0x18acf3d6, 0xce89e299, 0xb4a84fe0, 0xfd13e0b7, 0x7cc43b81, 0xd2ada8d9,  
            0x165fa266, 0x80957705, 0x93cc7314, 0x211a1477, 0xe6ad2065, 0x77b5fa86,  
            0xc75442f5, 0xfb9d35cf, 0xebcdaf0c, 0x7b3e89a0, 0xd6411bd3, 0xae1e7e49,  
            0x00250e2d, 0x2071b35e, 0x226800bb, 0x57b8e0af, 0x2464369b, 0xf009b91e,  
            0x5563911d, 0x59dfa6aa, 0x78c14389, 0xd95a537f, 0x207d5ba2, 0x02e5b9c5,  
            0x83260376, 0x6295cfa9, 0x11c81968, 0x4e734a41, 0xb3472dca, 0x7b14a94a,  
            0x1b510052, 0x9a532915, 0xd60f573f, 0xbc9bc6e4, 0x2b60a476, 0x81e67400,  
            0x08ba6fb5, 0x571be91f, 0xf296ec6b, 0x2a0dd915, 0xb6636521, 0xe7b9f9b6,  
            0xff34052e, 0xc5855664, 0x53b02d5d, 0xa99f8fa1, 0x08ba4799, 0x6e85076a,  
            0x4b7a70e9, 0xb5b32944, 0xdb75092e, 0xc4192623, 0xad6ea6b0, 0x49a7df7d,  
            0x9cee60b8, 0x8fedb266, 0xecaa8c71, 0x699a17ff, 0x5664526c, 0xc2b19ee1,  
            0x193602a5, 0x75094c29, 0xa0591340, 0xe4183a3e, 0x3f54989a, 0x5b429d65,  
            0x6b8fe4d6, 0x99f73fd6, 0xa1d29c07, 0xefe830f5, 0x4d2d38e6, 0xf0255dc1,  
            0x4cdd2086, 0x8470eb26, 0x6382e9c6, 0x021ecc5e, 0x09686b3f, 0x3ebaefc9,  
            0x3c971814, 0x6b6a70a1, 0x687f3584, 0x52a0e286, 0xb79c5305, 0xaa500737,  
            0x3e07841c, 0x7fdeae5c, 0x8e7d44ec, 0x5716f2b8, 0xb03ada37, 0xf0500c0d,  
            0xf01c1f04, 0x0200b3ff, 0xae0cf51a, 0x3cb574b2, 0x25837a58, 0xdc0921bd,  
            0xd19113f9, 0x7ca92ff6, 0x94324773, 0x22f54701, 0x3ae5e581, 0x37c2dadc,  
            0xc8b57634, 0x9af3dda7, 0xa9446146, 0x0fd0030e, 0xecc8c73e, 0xa4751e41,  
            0xe238cd99, 0x3bea0e2f, 0x3280bba1, 0x183eb331, 0x4e548b38, 0x4f6db908,  
            0x6f420d03, 0xf60a04bf, 0x2cb81290, 0x24977c79, 0x5679b072, 0xbcaf89af,  
            0xde9a771f, 0xd9930810, 0xb38bae12, 0xdccf3f2e, 0x5512721f, 0x2e6b7124,  
            0x501adde6, 0x9f84cd87, 0x7a584718, 0x7408da17, 0xbc9f9abc, 0xe94b7d8c,  
            0xec7aec3a, 0xdb851dfa, 0x63094366, 0xc464c3d2, 0xef1c1847, 0x3215d908,  
            0xdd433b37, 0x24c2ba16, 0x12a14d43, 0x2a65c451, 0x50940002, 0x133ae4dd,  
            0x71dff89e, 0x10314e55, 0x81ac77d6, 0x5f11199b, 0x043556f1, 0xd7a3c76b,  
            0x3c11183b, 0x5924a509, 0xf28fe6ed, 0x97f1fbfa, 0x9ebabf2c, 0x1e153c6e,  
            0x86e34570, 0xeae96fb1, 0x860e5e0a, 0x5a3e2ab3, 0x771fe71c, 0x4e3d06fa,  
            0x2965dcb9, 0x99e71d0f, 0x803e89d6, 0x5266c825, 0x2e4cc978, 0x9c10b36a,  
            0xc6150eba, 0x94e2ea78, 0xa5fc3c53, 0x1e0a2df4, 0xf2f74ea7, 0x361d2b3d,  
            0x1939260f, 0x19c27960, 0x5223a708, 0xf71312b6, 0xebadfe6e, 0xeac31f66,  
            0xe3bc4595, 0xa67bc883, 0xb17f37d1, 0x018cff28, 0xc332ddef, 0xbe6c5aa5,  
            0x65582185, 0x68ab9802, 0xeecea50f, 0xdb2f953b, 0x2aef7dad, 0x5b6e2f84,  
            0x1521b628, 0x29076170, 0xecdd4775, 0x619f1510, 0x13cca830, 0xeb61bd96,  
            0x0334fe1e, 0xaa0363cf, 0xb5735c90, 0x4c70a239, 0xd59e9e0b, 0xcbaade14,  
            0xeecc86bc, 0x60622ca7, 0x9cab5cab, 0xb2f3846e, 0x648b1eaf, 0x19bdf0ca,  
            0xa02369b9, 0x655abb50, 0x40685a32, 0x3c2ab4b3, 0x319ee9d5, 0xc021b8f7,  
            0x9b540b19, 0x875fa099, 0x95f7997e, 0x623d7da8, 0xf837889a, 0x97e32d77,  
            0x11ed935f, 0x16681281, 0x0e358829, 0xc7e61fd6, 0x96dedfa1, 0x7858ba99,  
            0x57f584a5, 0x1b227263, 0x9b83c3ff, 0x1ac24696, 0xcdb30aeb, 0x532e3054,  
            0x8fd948e4, 0x6dbc3128, 0x58ebf2ef, 0x34c6ffea, 0xfe28ed61, 0xee7c3c73,  
            0x5d4a14d9, 0xe864b7e3, 0x42105d14, 0x203e13e0, 0x45eee2b6, 0xa3aaabea,  
            0xdb6c4f15, 0xfacb4fd0, 0xc742f442, 0xef6abbb5, 0x654f3b1d, 0x41cd2105,  
            0xd81e799e, 0x86854dc7, 0xe44b476a, 0x3d816250, 0xcf62a1f2, 0x5b8d2646,  
            0xfc8883a0, 0xc1c7b6a3, 0x7f1524c3, 0x69cb7492, 0x47848a0b, 0x5692b285,  
            0x095bbf00, 0xad19489d, 0x1462b174, 0x23820e00, 0x58428d2a, 0x0c55f5ea,  
            0x1dadf43e, 0x233f7061, 0x3372f092, 0x8d937e41, 0xd65fecf1, 0x6c223bdb,  
            0x7cde3759, 0xcbee7460, 0x4085f2a7, 0xce77326e, 0xa6078084, 0x19f8509e,  
            0xe8efd855, 0x61d99735, 0xa969a7aa, 0xc50c06c2, 0x5a04abfc, 0x800bcadc,  
            0x9e447a2e, 0xc3453484, 0xfdd56705, 0x0e1e9ec9, 0xdb73dbd3, 0x105588cd,  
            0x675fda79, 0xe3674340, 0xc5c43465, 0x713e38d8, 0x3d28f89e, 0xf16dff20,  
            0x153e21e7, 0x8fb03d4a, 0xe6e39f2b, 0xdb83adf7, 0xe93d5a68, 0x948140f7,  
            0xf64c261c, 0x94692934, 0x411520f7, 0x7602d4f7, 0xbcf46b2e, 0xd4a20068,  
            0xd4082471, 0x3320f46a, 0x43b7d4b7, 0x500061af, 0x1e39f62e, 0x97244546,  
            0x14214f74, 0xbf8b8840, 0x4d95fc1d, 0x96b591af, 0x70f4ddd3, 0x66a02f45,  
            0xbfbc09ec, 0x03bd9785, 0x7fac6dd0, 0x31cb8504, 0x96eb27b3, 0x55fd3941,  
            0xda2547e6, 0xabca0a9a, 0x28507825, 0x530429f4, 0x0a2c86da, 0xe9b66dfb,  
            0x68dc1462, 0xd7486900, 0x680ec0a4, 0x27a18dee, 0x4f3ffea2, 0xe887ad8c,  
            0xb58ce006, 0x7af4d6b6, 0xaace1e7c, 0xd3375fec, 0xce78a399, 0x406b2a42,  
            0x20fe9e35, 0xd9f385b9, 0xee39d7ab, 0x3b124e8b, 0x1dc9faf7, 0x4b6d1856,  
            0x26a36631, 0xeae397b2, 0x3a6efa74, 0xdd5b4332, 0x6841e7f7, 0xca7820fb,  
            0xfb0af54e, 0xd8feb397, 0x454056ac, 0xba489527, 0x55533a3a, 0x20838d87,  
            0xfe6ba9b7, 0xd096954b, 0x55a867bc, 0xa1159a58, 0xcca92963, 0x99e1db33,  
            0xa62a4a56, 0x3f3125f9, 0x5ef47e1c, 0x9029317c, 0xfdf8e802, 0x04272f70,  
            0x80bb155c, 0x05282ce3, 0x95c11548, 0xe4c66d22, 0x48c1133f, 0xc70f86dc,  
            0x07f9c9ee, 0x41041f0f, 0x404779a4, 0x5d886e17, 0x325f51eb, 0xd59bc0d1,  
            0xf2bcc18f, 0x41113564, 0x257b7834, 0x602a9c60, 0xdff8e8a3, 0x1f636c1b,  
            0x0e12b4c2, 0x02e1329e, 0xaf664fd1, 0xcad18115, 0x6b2395e0, 0x333e92e1,  
            0x3b240b62, 0xeebeb922, 0x85b2a20e, 0xe6ba0d99, 0xde720c8c, 0x2da2f728,  
            0xd0127845, 0x95b794fd, 0x647d0862, 0xe7ccf5f0, 0x5449a36f, 0x877d48fa,  
            0xc39dfd27, 0xf33e8d1e, 0x0a476341, 0x992eff74, 0x3a6f6eab, 0xf4f8fd37,  
            0xa812dc60, 0xa1ebddf8, 0x991be14c, 0xdb6e6b0d, 0xc67b5510, 0x6d672c37,  
            0x2765d43b, 0xdcd0e804, 0xf1290dc7, 0xcc00ffa3, 0xb5390f92, 0x690fed0b,  
            0x667b9ffb, 0xcedb7d9c, 0xa091cf0b, 0xd9155ea3, 0xbb132f88, 0x515bad24,  
            0x7b9479bf, 0x763bd6eb, 0x37392eb3, 0xcc115979, 0x8026e297, 0xf42e312d,  
            0x6842ada7, 0xc66a2b3b, 0x12754ccc, 0x782ef11c, 0x6a124237, 0xb79251e7,  
            0x06a1bbe6, 0x4bfb6350, 0x1a6b1018, 0x11caedfa, 0x3d25bdd8, 0xe2e1c3c9,  
            0x44421659, 0x0a121386, 0xd90cec6e, 0xd5abea2a, 0x64af674e, 0xda86a85f,  
            0xbebfe988, 0x64e4c3fe, 0x9dbc8057, 0xf0f7c086, 0x60787bf8, 0x6003604d,  
            0xd1fd8346, 0xf6381fb0, 0x7745ae04, 0xd736fccc, 0x83426b33, 0xf01eab71,  
            0xb0804187, 0x3c005e5f, 0x77a057be, 0xbde8ae24, 0x55464299, 0xbf582e61,  
            0x4e58f48f, 0xf2ddfda2, 0xf474ef38, 0x8789bdc2, 0x5366f9c3, 0xc8b38e74,  
            0xb475f255, 0x46fcd9b9, 0x7aeb2661, 0x8b1ddf84, 0x846a0e79, 0x915f95e2,  
            0x466e598e, 0x20b45770, 0x8cd55591, 0xc902de4c, 0xb90bace1, 0xbb8205d0,  
            0x11a86248, 0x7574a99e, 0xb77f19b6, 0xe0a9dc09, 0x662d09a1, 0xc4324633,  
            0xe85a1f02, 0x09f0be8c, 0x4a99a025, 0x1d6efe10, 0x1ab93d1d, 0x0ba5a4df,  
            0xa186f20f, 0x2868f169, 0xdcb7da83, 0x573906fe, 0xa1e2ce9b, 0x4fcd7f52,  
            0x50115e01, 0xa70683fa, 0xa002b5c4, 0x0de6d027, 0x9af88c27, 0x773f8641,  
            0xc3604c06, 0x61a806b5, 0xf0177a28, 0xc0f586e0, 0x006058aa, 0x30dc7d62,  
            0x11e69ed7, 0x2338ea63, 0x53c2dd94, 0xc2c21634, 0xbbcbee56, 0x90bcb6de,  
            0xebfc7da1, 0xce591d76, 0x6f05e409, 0x4b7c0188, 0x39720a3d, 0x7c927c24,  
            0x86e3725f, 0x724d9db9, 0x1ac15bb4, 0xd39eb8fc, 0xed545578, 0x08fca5b5,  
            0xd83d7cd3, 0x4dad0fc4, 0x1e50ef5e, 0xb161e6f8, 0xa28514d9, 0x6c51133c,  
            0x6fd5c7e7, 0x56e14ec4, 0x362abfce, 0xddc6c837, 0xd79a3234, 0x92638212,  
            0x670efa8e, 0x406000e0, 0x3a39ce37, 0xd3faf5cf, 0xabc27737, 0x5ac52d1b,  
            0x5cb0679e, 0x4fa33742, 0xd3822740, 0x99bc9bbe, 0xd5118e9d, 0xbf0f7315,  
            0xd62d1c7e, 0xc700c47b, 0xb78c1b6b, 0x21a19045, 0xb26eb1be, 0x6a366eb4,  
            0x5748ab2f, 0xbc946e79, 0xc6a376d2, 0x6549c2c8, 0x530ff8ee, 0x468dde7d,  
            0xd5730a1d, 0x4cd04dc6, 0x2939bbdb, 0xa9ba4650, 0xac9526e8, 0xbe5ee304,  
            0xa1fad5f0, 0x6a2d519a, 0x63ef8ce2, 0x9a86ee22, 0xc089c2b8, 0x43242ef6,  
            0xa51e03aa, 0x9cf2d0a4, 0x83c061ba, 0x9be96a4d, 0x8fe51550, 0xba645bd6,  
            0x2826a2f9, 0xa73a3ae1, 0x4ba99586, 0xef5562e9, 0xc72fefd3, 0xf752f7da,  
            0x3f046f69, 0x77fa0a59, 0x80e4a915, 0x87b08601, 0x9b09e6ad, 0x3b3ee593,  
            0xe990fd5a, 0x9e34d797, 0x2cf0b7d9, 0x022b8b51, 0x96d5ac3a, 0x017da67d,  
            0xd1cf3ed6, 0x7c7d2d28, 0x1f9f25cf, 0xadf2b89b, 0x5ad6b472, 0x5a88f54c,  
            0xe029ac71, 0xe019a5e6, 0x47b0acfd, 0xed93fa9b, 0xe8d3c48d, 0x283b57cc,  
            0xf8d56629, 0x79132e28, 0x785f0191, 0xed756055, 0xf7960e44, 0xe3d35e8c,  
            0x15056dd4, 0x88f46dba, 0x03a16125, 0x0564f0bd, 0xc3eb9e15, 0x3c9057a2,  
            0x97271aec, 0xa93a072a, 0x1b3f6d9b, 0x1e6321f5, 0xf59c66fb, 0x26dcf319,  
            0x7533d928, 0xb155fdf5, 0x03563482, 0x8aba3cbb, 0x28517711, 0xc20ad9f8,  
            0xabcc5167, 0xccad925f, 0x4de81751, 0x3830dc8e, 0x379d5862, 0x9320f991,  
            0xea7a90c2, 0xfb3e7bce, 0x5121ce64, 0x774fbe32, 0xa8b6e37e, 0xc3293d46,  
            0x48de5369, 0x6413e680, 0xa2ae0810, 0xdd6db224, 0x69852dfd, 0x09072166,  
            0xb39a460a, 0x6445c0dd, 0x586cdecf, 0x1c20c8ae, 0x5bbef7dd, 0x1b588d40,  
            0xccd2017f, 0x6bb4e3bb, 0xdda26a7e, 0x3a59ff45, 0x3e350a44, 0xbcb4cdd5,  
            0x72eacea8, 0xfa6484bb, 0x8d6612ae, 0xbf3c6f47, 0xd29be463, 0x542f5d9e,  
            0xaec2771b, 0xf64e6370, 0x740e0d8d, 0xe75b1357, 0xf8721671, 0xaf537d5d,  
            0x4040cb08, 0x4eb4e2cc, 0x34d2466a, 0x0115af84, 0xe1b00428, 0x95983a1d,  
            0x06b89fb4, 0xce6ea048, 0x6f3f3b82, 0x3520ab82, 0x011a1d4b, 0x277227f8,  
            0x611560b1, 0xe7933fdc, 0xbb3a792b, 0x344525bd, 0xa08839e1, 0x51ce794b,  
            0x2f32c9b7, 0xa01fbac9, 0xe01cc87e, 0xbcc7d1f6, 0xcf0111c3, 0xa1e8aac7,  
            0x1a908749, 0xd44fbd9a, 0xd0dadecb, 0xd50ada38, 0x0339c32a, 0xc6913667,  
            0x8df9317c, 0xe0b12b4f, 0xf79e59b7, 0x43f5bb3a, 0xf2d519ff, 0x27d9459c,  
            0xbf97222c, 0x15e6fc2a, 0x0f91fc71, 0x9b941525, 0xfae59361, 0xceb69ceb,  
            0xc2a86459, 0x12baa8d1, 0xb6c1075e, 0xe3056a0c, 0x10d25065, 0xcb03a442,  
            0xe0ec6e0e, 0x1698db3b, 0x4c98a0be, 0x3278e964, 0x9f1f9532, 0xe0d392df,  
            0xd3a0342b, 0x8971f21e, 0x1b0a7441, 0x4ba3348c, 0xc5be7120, 0xc37632d8,  
            0xdf359f8d, 0x9b992f2e, 0xe60b6f47, 0x0fe3f11d, 0xe54cda54, 0x1edad891,  
            0xce6279cf, 0xcd3e7e6f, 0x1618b166, 0xfd2c1d05, 0x848fd2c5, 0xf6fb2299,  
            0xf523f357, 0xa6327623, 0x93a83531, 0x56cccd02, 0xacf08162, 0x5a75ebb5,  
            0x6e163697, 0x88d273cc, 0xde966292, 0x81b949d0, 0x4c50901b, 0x71c65614,  
            0xe6c6c7bd, 0x327a140a, 0x45e1d006, 0xc3f27b9a, 0xc9aa53fd, 0x62a80f00,  
            0xbb25bfe2, 0x35bdd2f6, 0x71126905, 0xb2040222, 0xb6cbcf7c, 0xcd769c2b,  
            0x53113ec0, 0x1640e3d3, 0x38abbd60, 0x2547adf0, 0xba38209c, 0xf746ce76,  
            0x77afa1c5, 0x20756060, 0x85cbfe4e, 0x8ae88dd8, 0x7aaaf9b0, 0x4cf9aa7e,  
            0x1948c25c, 0x02fb8a8c, 0x01c36ae4, 0xd6ebe1f9, 0x90d4f869, 0xa65cdea0,  
            0x3f09252d, 0xc208e69f, 0xb74e6132, 0xce77e25b, 0x578fdfe3, 0x3ac372e6};  
    // bcrypt IV: "OrpheanBeholderScryDoubt"  
    static private final int bf\_crypt\_ciphertext[] = {0x4f727068, 0x65616e42,  
            0x65686f6c, 0x64657253, 0x63727944, 0x6f756274};  
    // Table for Base64 encoding  
    static private final char base64\_code[] = {'.', '/', 'A', 'B', 'C', 'D', 'E', 'F',  
            'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U',  
            'V', 'W', 'X', 'Y', 'Z', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j',  
            'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y',  
            'z', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'};  
    // Table for Base64 decoding  
    static private final byte index\_64[] = {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,  
            -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,  
            -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 0, 1, 54, 55,  
            56, 57, 58, 59, 60, 61, 62, 63, -1, -1, -1, -1, -1, -1, -1, 2, 3, 4, 5, 6, 7,  
            8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27,  
            -1, -1, -1, -1, -1, -1, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,  
            41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, -1, -1, -1, -1, -1};  
    // Expanded Blowfish key  
    private int P[];  
    private int S[];  
  
    /\*\*  
     \* Encode a byte array using bcrypt's slightly-modified base64 encoding scheme. Note  
     \* that this is <strong>not<> compatible with the standard MIME-base64  
     \* encoding.  
     \*  
     \* @param d   the byte array to encode  
     \* @param len the number of bytes to encode  
     \* @param rs  the destination buffer for the base64-encoded string  
     \* @throws IllegalArgumentException if the length is invalid  
     \*/  
    static void encode\_base64(byte d[], int len, StringBuilder rs)  
            throws IllegalArgumentException {  
        int off = 0;  
        int c1, c2;  
  
        if (len <= 0 || len > d.length) {  
            throw new IllegalArgumentException("Invalid len");  
        }  
  
        while (off < len) {  
            c1 = d[off++] & 0xff;  
            rs.append(base64\_code[(c1 >> 2) & 0x3f]);  
            c1 = (c1 & 0x03) << 4;  
            if (off >= len) {  
                rs.append(base64\_code[c1 & 0x3f]);  
                break;  
            }  
            c2 = d[off++] & 0xff;  
            c1 |= (c2 >> 4) & 0x0f;  
            rs.append(base64\_code[c1 & 0x3f]);  
            c1 = (c2 & 0x0f) << 2;  
            if (off >= len) {  
                rs.append(base64\_code[c1 & 0x3f]);  
                break;  
            }  
            c2 = d[off++] & 0xff;  
            c1 |= (c2 >> 6) & 0x03;  
            rs.append(base64\_code[c1 & 0x3f]);  
            rs.append(base64\_code[c2 & 0x3f]);  
        }  
    }  
  
    /\*\*  
     \* Look up the 3 bits base64-encoded by the specified character, range-checking  
     \* against conversion table  
     \*  
     \* @param x the base64-encoded value  
     \* @return the decoded value of x  
     \*/  
    private static byte char64(char x) {  
        if (x > index\_64.length) {  
            return -1;  
        }  
        return index\_64[x];  
    }  
  
    /\*\*  
     \* Decode a string encoded using bcrypt's base64 scheme to a byte array. Note that  
     \* this is \*not\* compatible with the standard MIME-base64 encoding.  
     \*  
     \* @param s       the string to decode  
     \* @param maxolen the maximum number of bytes to decode  
     \* @return an array containing the decoded bytes  
     \* @throws IllegalArgumentException if maxolen is invalid  
     \*/  
    static byte[] decode\_base64(String s, int maxolen) throws IllegalArgumentException {  
        ByteArrayOutputStream out = new ByteArrayOutputStream(maxolen);  
        int off = 0, slen = s.length(), olen = 0;  
        byte c1, c2, c3, c4, o;  
  
        if (maxolen <= 0) {  
            throw new IllegalArgumentException("Invalid maxolen");  
        }  
  
        while (off < slen - 1 && olen < maxolen) {  
            c1 = char64(s.charAt(off++));  
            c2 = char64(s.charAt(off++));  
            if (c1 == -1 || c2 == -1) {  
                break;  
            }  
            o = (byte) (c1 << 2);  
            o |= (c2 & 0x30) >> 4;  
            out.write(o);  
            if (++olen >= maxolen || off >= slen) {  
                break;  
            }  
            c3 = char64(s.charAt(off++));  
            if (c3 == -1) {  
                break;  
            }  
            o = (byte) ((c2 & 0x0f) << 4);  
            o |= (c3 & 0x3c) >> 2;  
            out.write(o);  
            if (++olen >= maxolen || off >= slen) {  
                break;  
            }  
            c4 = char64(s.charAt(off++));  
            o = (byte) ((c3 & 0x03) << 6);  
            o |= c4;  
            out.write(o);  
            ++olen;  
        }  
  
        return out.toByteArray();  
    }  
  
    /\*\*  
     \* Cycically extract a word of key material  
     \*  
     \* @param data the string to extract the data from  
     \* @param offp a "pointer" (as a one-entry array) to the current offset into data  
     \* @return the next word of material from data  
     \*/  
    private static int streamtoword(byte data[], int offp[]) {  
        int i;  
        int word = 0;  
        int off = offp[0];  
  
        for (i = 0; i < 4; i++) {  
            word = (word << 8) | (data[off] & 0xff);  
            off = (off + 1) % data.length;  
        }  
  
        offp[0] = off;  
        return word;  
    }  
  
    static long roundsForLogRounds(int log\_rounds) {  
        if (log\_rounds < 4 || log\_rounds > 31) {  
            throw new IllegalArgumentException("Bad number of rounds");  
        }  
        return 1L << log\_rounds;  
    }  
  
    /\*\*  
     \* Hash a password using the OpenBSD bcrypt scheme  
     \*  
     \* @param password the password to hash  
     \* @param salt     the salt to hash with (perhaps generated using BCrypt.gensalt)  
     \* @return the hashed password  
     \* @throws IllegalArgumentException if invalid salt is passed  
     \*/  
    public static String hashpw(String password, String salt) throws IllegalArgumentException {  
        BCrypt B;  
        String real\_salt;  
        byte passwordb[], saltb[], hashed[];  
        char minor = (char) 0;  
        int rounds, off = 0;  
        StringBuilder rs = new StringBuilder();  
  
        if (salt == null) {  
            throw new IllegalArgumentException("salt cannot be null");  
        }  
  
        int saltLength = salt.length();  
  
        if (saltLength < 28) {  
            throw new IllegalArgumentException("Invalid salt");  
        }  
  
        if (salt.charAt(0) != '$' || salt.charAt(1) != '2') {  
            throw new IllegalArgumentException("Invalid salt version");  
        }  
        if (salt.charAt(2) == '$') {  
            off = 3;  
        } else {  
            minor = salt.charAt(2);  
            if (minor != 'a' || salt.charAt(3) != '$') {  
                throw new IllegalArgumentException("Invalid salt revision");  
            }  
            off = 4;  
        }  
  
        if (saltLength - off < 25) {  
            throw new IllegalArgumentException("Invalid salt");  
        }  
  
        // Extract number of rounds  
        if (salt.charAt(off + 2) > '$') {  
            throw new IllegalArgumentException("Missing salt rounds");  
        }  
        rounds = Integer.parseInt(salt.substring(off, off + 2));  
  
        real\_salt = salt.substring(off + 3, off + 25);  
        try {  
            passwordb = (password + (minor >= 'a' ? "\000" : "")).getBytes("UTF-8");  
        } catch (UnsupportedEncodingException uee) {  
            throw new AssertionError("UTF-8 is not supported");  
        }  
  
        saltb = decode\_base64(real\_salt, BCRYPT\_SALT\_LEN);  
  
        B = new BCrypt();  
        hashed = B.crypt\_raw(passwordb, saltb, rounds);  
  
        rs.append("$2");  
        if (minor >= 'a') {  
            rs.append(minor);  
        }  
        rs.append("$");  
        if (rounds < 10) {  
            rs.append("0");  
        }  
        rs.append(rounds);  
        rs.append("$");  
        encode\_base64(saltb, saltb.length, rs);  
        encode\_base64(hashed, bf\_crypt\_ciphertext.length \* 4 - 1, rs);  
        return rs.toString();  
    }  
  
    /\*\*  
     \* Generate a salt for use with the BCrypt.hashpw() method  
     \*  
     \* @param log\_rounds the log2 of the number of rounds of hashing to apply - the work  
     \*                   factor therefore increases as 2\*\*log\_rounds. Minimum 4, maximum 31.  
     \* @param random     an instance of SecureRandom to use  
     \* @return an encoded salt value  
     \*/  
    public static String gensalt(int log\_rounds, SecureRandom random) {  
        if (log\_rounds < MIN\_LOG\_ROUNDS || log\_rounds > MAX\_LOG\_ROUNDS) {  
            throw new IllegalArgumentException("Bad number of rounds");  
        }  
        StringBuilder rs = new StringBuilder();  
        byte rnd[] = new byte[BCRYPT\_SALT\_LEN];  
  
        random.nextBytes(rnd);  
  
        rs.append("$2a$");  
        if (log\_rounds < 10) {  
            rs.append("0");  
        }  
        rs.append(log\_rounds);  
        rs.append("$");  
        encode\_base64(rnd, rnd.length, rs);  
        return rs.toString();  
    }  
  
    /\*\*  
     \* Generate a salt for use with the BCrypt.hashpw() method  
     \*  
     \* @param log\_rounds the log2 of the number of rounds of hashing to apply - the work  
     \*                   factor therefore increases as 2\*\*log\_rounds. Minimum 4, maximum 31.  
     \* @return an encoded salt value  
     \*/  
    public static String gensalt(int log\_rounds) {  
        return gensalt(log\_rounds, new SecureRandom());  
    }  
  
    /\*\*  
     \* Generate a salt for use with the BCrypt.hashpw() method, selecting a reasonable  
     \* default for the number of hashing rounds to apply  
     \*  
     \* @return an encoded salt value  
     \*/  
    public static String gensalt() {  
        return gensalt(GENSALT\_DEFAULT\_LOG2\_ROUNDS);  
    }  
  
    /\*\*  
     \* Check that a plaintext password matches a previously hashed one  
     \*  
     \* @param plaintext the plaintext password to verify  
     \* @param hashed    the previously-hashed password  
     \* @return true if the passwords match, false otherwise  
     \*/  
    public static boolean checkpw(String plaintext, String hashed) {  
        return equalsNoEarlyReturn(hashed, hashpw(plaintext, hashed));  
    }  
  
    static boolean equalsNoEarlyReturn(String a, String b) {  
        char[] caa = a.toCharArray();  
        char[] cab = b.toCharArray();  
  
        if (caa.length != cab.length) {  
            return false;  
        }  
  
        byte ret = 0;  
        for (int i = 0; i < caa.length; i++) {  
            ret |= caa[i] ^ cab[i];  
        }  
        return ret == 0;  
    }  
  
    /\*\*  
     \* Blowfish encipher a single 64-bit block encoded as two 32-bit halves  
     \*  
     \* @param lr  an array containing the two 32-bit half blocks  
     \* @param off the position in the array of the blocks  
     \*/  
    private final void encipher(int lr[], int off) {  
        int i, n, l = lr[off], r = lr[off + 1];  
  
        l ^= P[0];  
        for (i = 0; i <= BLOWFISH\_NUM\_ROUNDS - 2; ) {  
            // Feistel substitution on left word  
            n = S[(l >> 24) & 0xff];  
            n += S[0x100 | ((l >> 16) & 0xff)];  
            n ^= S[0x200 | ((l >> 8) & 0xff)];  
            n += S[0x300 | (l & 0xff)];  
            r ^= n ^ P[++i];  
  
            // Feistel substitution on right word  
            n = S[(r >> 24) & 0xff];  
            n += S[0x100 | ((r >> 16) & 0xff)];  
            n ^= S[0x200 | ((r >> 8) & 0xff)];  
            n += S[0x300 | (r & 0xff)];  
            l ^= n ^ P[++i];  
        }  
        lr[off] = r ^ P[BLOWFISH\_NUM\_ROUNDS + 1];  
        lr[off + 1] = l;  
    }  
  
    /\*\*  
     \* Initialise the Blowfish key schedule  
     \*/  
    private void init\_key() {  
        P = (int[]) P\_orig.clone();  
        S = (int[]) S\_orig.clone();  
    }  
  
    /\*\*  
     \* Key the Blowfish cipher  
     \*  
     \* @param key an array containing the key  
     \*/  
    private void key(byte key[]) {  
        int i;  
        int koffp[] = {0};  
        int lr[] = {0, 0};  
        int plen = P.length, slen = S.length;  
  
        for (i = 0; i < plen; i++) {  
            P[i] = P[i] ^ streamtoword(key, koffp);  
        }  
  
        for (i = 0; i < plen; i += 2) {  
            encipher(lr, 0);  
            P[i] = lr[0];  
            P[i + 1] = lr[1];  
        }  
  
        for (i = 0; i < slen; i += 2) {  
            encipher(lr, 0);  
            S[i] = lr[0];  
            S[i + 1] = lr[1];  
        }  
    }  
  
    /\*\*  
     \* Perform the "enhanced key schedule" step described by Provos and Mazieres in  
     \* "A Future-Adaptable Password Scheme" http://www.openbsd.org/papers/bcrypt-paper.ps  
     \*  
     \* @param data salt information  
     \* @param key  password information  
     \*/  
    private void ekskey(byte data[], byte key[]) {  
        int i;  
        int koffp[] = {0}, doffp[] = {0};  
        int lr[] = {0, 0};  
        int plen = P.length, slen = S.length;  
  
        for (i = 0; i < plen; i++) {  
            P[i] = P[i] ^ streamtoword(key, koffp);  
        }  
  
        for (i = 0; i < plen; i += 2) {  
            lr[0] ^= streamtoword(data, doffp);  
            lr[1] ^= streamtoword(data, doffp);  
            encipher(lr, 0);  
            P[i] = lr[0];  
            P[i + 1] = lr[1];  
        }  
  
        for (i = 0; i < slen; i += 2) {  
            lr[0] ^= streamtoword(data, doffp);  
            lr[1] ^= streamtoword(data, doffp);  
            encipher(lr, 0);  
            S[i] = lr[0];  
            S[i + 1] = lr[1];  
        }  
    }  
  
    /\*\*  
     \* Perform the central password hashing step in the bcrypt scheme  
     \*  
     \* @param password   the password to hash  
     \* @param salt       the binary salt to hash with the password  
     \* @param log\_rounds the binary logarithm of the number of rounds of hashing to apply  
     \* @return an array containing the binary hashed password  
     \*/  
    private byte[] crypt\_raw(byte password[], byte salt[], int log\_rounds) {  
        int cdata[] = (int[]) bf\_crypt\_ciphertext.clone();  
        int clen = cdata.length;  
        byte ret[];  
  
        long rounds = roundsForLogRounds(log\_rounds);  
  
        init\_key();  
        ekskey(salt, password);  
        for (long i = 0; i < rounds; i++) {  
            key(password);  
            key(salt);  
        }  
  
        for (int i = 0; i < 64; i++) {  
            for (int j = 0; j < (clen >> 1); j++) {  
                encipher(cdata, j << 1);  
            }  
        }  
  
        ret = new byte[clen \* 4];  
        for (int i = 0, j = 0; i < clen; i++) {  
            ret[j++] = (byte) ((cdata[i] >> 24) & 0xff);  
            ret[j++] = (byte) ((cdata[i] >> 16) & 0xff);  
            ret[j++] = (byte) ((cdata[i] >> 8) & 0xff);  
            ret[j++] = (byte) (cdata[i] & 0xff);  
        }  
        return ret;  
    }  
}

package org.linlinjava.litemall.core.util.bcrypt;

/\*

\* Copyright 2002-2011 the original author or authors.

\*

\* Licensed under the Apache License, Version 2.0 (the "License");

\* you may not use this file except in compliance with the License.

\* You may obtain a copy of the License at

\*

\* http://www.apache.org/licenses/LICENSE-2.0

\*

\* Unless required by applicable law or agreed to in writing, software

\* distributed under the License is distributed on an "AS IS" BASIS,

\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

\* See the License for the specific language governing permissions and

\* limitations under the License.

\*/

import org.apache.commons.logging.Log;

import org.apache.commons.logging.LogFactory;

import java.security.SecureRandom;

import java.util.regex.Pattern;

/\*\*

\* Implementation of PasswordEncoder that uses the BCrypt strong hashing function. Clients

\* can optionally supply a "strength" (a.k.a. log rounds in BCrypt) and a SecureRandom

\* instance. The larger the strength parameter the more work will have to be done

\* (exponentially) to hash the passwords. The default value is 10.

\*

\* @author Dave Syer

\*/

public class BCryptPasswordEncoder {

private final Log logger = LogFactory.getLog(getClass());

private final int strength;

private final SecureRandom random;

private Pattern BCRYPT\_PATTERN = Pattern

.compile("\\A\\$2a?\\$\\d\\d\\$[./0-9A-Za-z]{53}");

public BCryptPasswordEncoder() {

this(-1);

}

/\*\*

\* @param strength the log rounds to use, between 4 and 31

\*/

public BCryptPasswordEncoder(int strength) {

this(strength, null);

}

/\*\*

\* @param strength the log rounds to use, between 4 and 31

\* @param random the secure random instance to use

\*/

public BCryptPasswordEncoder(int strength, SecureRandom random) {

if (strength != -1 && (strength < BCrypt.MIN\_LOG\_ROUNDS || strength > BCrypt.MAX\_LOG\_ROUNDS)) {

throw new IllegalArgumentException("Bad strength");

}

this.strength = strength;

this.random = random;

}

public String encode(CharSequence rawPassword) {

String salt;

if (strength > 0) {

if (random != null) {

salt = BCrypt.gensalt(strength, random);

} else {

salt = BCrypt.gensalt(strength);

}

} else {

salt = BCrypt.gensalt();

}

return BCrypt.hashpw(rawPassword.toString(), salt);

}

public boolean matches(CharSequence rawPassword, String encodedPassword) {

if (encodedPassword == null || encodedPassword.length() == 0) {

logger.warn("Empty encoded password");

return false;

}

if (!BCRYPT\_PATTERN.matcher(encodedPassword).matches()) {

logger.warn("Encoded password does not look like BCrypt");

return false;

}

return BCrypt.checkpw(rawPassword.toString(), encodedPassword);

}

}

package org.linlinjava.litemall.core.util;

import java.util.Random;

public class CharUtil {

public static String getRandomString(Integer num) {

String base = "abcdefghijklmnopqrstuvwxyz0123456789";

Random random = new Random();

StringBuffer sb = new StringBuffer();

for (int i = 0; i < num; i++) {

int number = random.nextInt(base.length());

sb.append(base.charAt(number));

}

return sb.toString();

}

public static String getRandomNum(Integer num) {

String base = "0123456789";

Random random = new Random();

StringBuffer sb = new StringBuffer();

for (int i = 0; i < num; i++) {

int number = random.nextInt(base.length());

sb.append(base.charAt(number));

}

return sb.toString();

}

}

package org.linlinjava.litemall.core.util;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

/\*\*

\* 日期格式化工具类

\*/

public class DateTimeUtil {

/\*\*

\* 格式 yyyy年MM月dd日 HH:mm:ss

\*

\* @param dateTime

\* @return

\*/

public static String getDateTimeDisplayString(LocalDateTime dateTime) {

DateTimeFormatter dtf2 = DateTimeFormatter.ofPattern("yyyy年MM月dd日 HH:mm:ss");

String strDate2 = dtf2.format(dateTime);

return strDate2;

}

}

package org.linlinjava.litemall.core.util;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.OutputStreamWriter;

import java.net.HttpURLConnection;

import java.net.URL;

import java.util.Map;

/\*\*

\* 向指定 URL 发送POST方法的请求

\*

\* @return 远程资源的响应结果

\*/

public class HttpUtil {

/\*\*

\* 向指定 URL 发送POST方法的请求

\*

\* @param url 发送请求的 URL

\* @param params 请求的参数集合

\* @return 远程资源的响应结果

\*/

@SuppressWarnings("unused")

public static String sendPost(String url, Map<String, String> params) {

OutputStreamWriter out = null;

BufferedReader in = null;

StringBuilder result = new StringBuilder();

try {

URL realUrl = new URL(url);

HttpURLConnection conn = (HttpURLConnection) realUrl.openConnection();

// 发送POST请求必须设置如下两行

conn.setDoOutput(true);

conn.setDoInput(true);

// POST方法

conn.setRequestMethod("POST");

// 设置通用的请求属性

conn.setRequestProperty("accept", "\*/\*");

conn.setRequestProperty("connection", "Keep-Alive");

conn.setRequestProperty("user-agent",

"Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1;SV1)");

conn.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");

conn.connect();

// 获取URLConnection对象对应的输出流

out = new OutputStreamWriter(conn.getOutputStream(), "UTF-8");

// 发送请求参数

if (params != null) {

StringBuilder param = new StringBuilder();

for (Map.Entry<String, String> entry : params.entrySet()) {

if (param.length() > 0) {

param.append("&");

}

param.append(entry.getKey());

param.append("=");

param.append(entry.getValue());

//System.out.println(entry.getKey()+":"+entry.getValue());

}

//System.out.println("param:"+param.toString());

out.write(param.toString());

}

// flush输出流的缓冲

out.flush();

// 定义BufferedReader输入流来读取URL的响应

in = new BufferedReader(

new InputStreamReader(conn.getInputStream(), "UTF-8"));

String line;

while ((line = in.readLine()) != null) {

result.append(line);

}

} catch (Exception e) {

e.printStackTrace();

}

//使用finally块来关闭输出流、输入流

finally {

try {

if (out != null) {

out.close();

}

if (in != null) {

in.close();

}

} catch (IOException ex) {

ex.printStackTrace();

}

}

return result.toString();

}

}

package org.linlinjava.litemall.core.util;

import javax.servlet.http.HttpServletRequest;

import java.net.InetAddress;

import java.net.UnknownHostException;

/\*\*

\* IP地址相关工具类

\*/

public class IpUtil {

public static String getIpAddr(HttpServletRequest request) {

String ipAddress = null;

try {

ipAddress = request.getHeader("x-forwarded-for");

if (ipAddress == null || ipAddress.length() == 0 || "unknown".equalsIgnoreCase(ipAddress)) {

ipAddress = request.getHeader("Proxy-Client-IP");

}

if (ipAddress == null || ipAddress.length() == 0 || "unknown".equalsIgnoreCase(ipAddress)) {

ipAddress = request.getHeader("WL-Proxy-Client-IP");

}

if (ipAddress == null || ipAddress.length() == 0 || "unknown".equalsIgnoreCase(ipAddress)) {

ipAddress = request.getRemoteAddr();

if (ipAddress.equals("127.0.0.1")) {

// 根据网卡取本机配置的IP

InetAddress inet = null;

try {

inet = InetAddress.getLocalHost();

} catch (UnknownHostException e) {

e.printStackTrace();

}

ipAddress = inet.getHostAddress();

}

}

// 对于通过多个代理的情况，第一个IP为客户端真实IP,多个IP按照','分割

if (ipAddress != null && ipAddress.length() > 15) { // "\*\*\*.\*\*\*.\*\*\*.\*\*\*".length()

// = 15

if (ipAddress.indexOf(",") > 0) {

ipAddress = ipAddress.substring(0, ipAddress.indexOf(","));

}

}

} catch (Exception e) {

ipAddress = "";

}

// ipAddress = this.getRequest().getRemoteAddr();

return ipAddress;

}

}

package org.linlinjava.litemall.core.util;

import com.fasterxml.jackson.core.type.TypeReference;

import com.fasterxml.jackson.databind.JsonNode;

import com.fasterxml.jackson.databind.ObjectMapper;

import java.io.IOException;

import java.util.List;

import java.util.Map;

public class JacksonUtil {

public static String parseString(String body, String field) {

ObjectMapper mapper = new ObjectMapper();

JsonNode node = null;

try {

node = mapper.readTree(body);

JsonNode leaf = node.get(field);

if (leaf != null)

return leaf.asText();

} catch (IOException e) {

e.printStackTrace();

}

return null;

}

public static List<String> parseStringList(String body, String field) {

ObjectMapper mapper = new ObjectMapper();

JsonNode node = null;

try {

node = mapper.readTree(body);

JsonNode leaf = node.get(field);

if (leaf != null)

return mapper.convertValue(leaf, new TypeReference<List<String>>() {

});

} catch (IOException e) {

e.printStackTrace();

}

return null;

}

public static Integer parseInteger(String body, String field) {

ObjectMapper mapper = new ObjectMapper();

JsonNode node = null;

try {

node = mapper.readTree(body);

JsonNode leaf = node.get(field);

if (leaf != null)

return leaf.asInt();

} catch (IOException e) {

e.printStackTrace();

}

return null;

}

public static List<Integer> parseIntegerList(String body, String field) {

ObjectMapper mapper = new ObjectMapper();

JsonNode node = null;

try {

node = mapper.readTree(body);

JsonNode leaf = node.get(field);

if (leaf != null)

return mapper.convertValue(leaf, new TypeReference<List<Integer>>() {

});

} catch (IOException e) {

e.printStackTrace();

}

return null;

}

public static Boolean parseBoolean(String body, String field) {

ObjectMapper mapper = new ObjectMapper();

JsonNode node = null;

try {

node = mapper.readTree(body);

JsonNode leaf = node.get(field);

if (leaf != null)

return leaf.asBoolean();

} catch (IOException e) {

e.printStackTrace();

}

return null;

}

public static Short parseShort(String body, String field) {

ObjectMapper mapper = new ObjectMapper();

JsonNode node = null;

try {

node = mapper.readTree(body);

JsonNode leaf = node.get(field);

if (leaf != null) {

Integer value = leaf.asInt();

return value.shortValue();

}

} catch (IOException e) {

e.printStackTrace();

}

return null;

}

public static Byte parseByte(String body, String field) {

ObjectMapper mapper = new ObjectMapper();

JsonNode node = null;

try {

node = mapper.readTree(body);

JsonNode leaf = node.get(field);

if (leaf != null) {

Integer value = leaf.asInt();

return value.byteValue();

}

} catch (IOException e) {

e.printStackTrace();

}

return null;

}

public static <T> T parseObject(String body, String field, Class<T> clazz) {

ObjectMapper mapper = new ObjectMapper();

JsonNode node = null;

try {

node = mapper.readTree(body);

node = node.get(field);

return mapper.treeToValue(node, clazz);

} catch (IOException e) {

e.printStackTrace();

}

return null;

}

public static Object toNode(String json) {

if (json == null) {

return null;

}

ObjectMapper mapper = new ObjectMapper();

try {

JsonNode jsonNode = mapper.readTree(json);

return jsonNode;

} catch (IOException e) {

e.printStackTrace();

}

return null;

}

public static Map<String, String> toMap(String data) {

ObjectMapper objectMapper = new ObjectMapper();

try {

return objectMapper.readValue(data, new TypeReference<Map<String, String>>(){});

} catch (IOException e) {

e.printStackTrace();

}

return null;

}

}

package org.linlinjava.litemall.core.util;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

/\*\*

\* RegexUtil类的代码是来自[AndroidUtilCode](https://github.com/Blankj/AndroidUtilCode)的RegexUtils类和RegexConstants类

\* https://github.com/Blankj/AndroidUtilCode/blob/master/utilcode/src/main/java/com/blankj/utilcode/util/RegexUtils.java

\* https://github.com/Blankj/AndroidUtilCode/blob/master/utilcode/src/main/java/com/blankj/utilcode/constant/RegexConstants.java

\*/

public class RegexUtil {

/\*\*

\* Regex of simple mobile.

\*/

public static final String REGEX\_MOBILE\_SIMPLE = "^[1]\\d{10}$";

/\*\*

\* Regex of exact mobile.

\* <p>china mobile: 134(0-8), 135, 136, 137, 138, 139, 147, 150, 151, 152, 157, 158, 159, 178, 182, 183, 184, 187, 188, 198</p>

\* <p>china unicom: 130, 131, 132, 145, 155, 156, 166, 171, 175, 176, 185, 186</p>

\* <p>china telecom: 133, 153, 173, 177, 180, 181, 189, 199</p>

\* <p>global star: 1349</p>

\* <p>virtual operator: 170</p>

\*/

public static final String REGEX\_MOBILE\_EXACT = "^((13[0-9])|(14[5,7])|(15[0-3,5-9])|(16[6])|(17[0,1,3,5-8])|(18[0-9])|(19[8,9]))\\d{8}$";

/\*\*

\* Regex of telephone number.

\*/

public static final String REGEX\_TEL = "^0\\d{2,3}[- ]?\\d{7,8}";

/\*\*

\* Regex of id card number which length is 15.

\*/

public static final String REGEX\_ID\_CARD15 = "^[1-9]\\d{7}((0\\d)|(1[0-2]))(([0|1|2]\\d)|3[0-1])\\d{3}$";

/\*\*

\* Regex of id card number which length is 18.

\*/

public static final String REGEX\_ID\_CARD18 = "^[1-9]\\d{5}[1-9]\\d{3}((0\\d)|(1[0-2]))(([0|1|2]\\d)|3[0-1])\\d{3}([0-9Xx])$";

/\*\*

\* Regex of email.

\*/

public static final String REGEX\_EMAIL = "^\\w+([-+.]\\w+)\*@\\w+([-.]\\w+)\*\\.\\w+([-.]\\w+)\*$";

/\*\*

\* Regex of url.

\*/

public static final String REGEX\_URL = "[a-zA-z]+://[^\\s]\*";

/\*\*

\* Regex of Chinese character.

\*/

public static final String REGEX\_ZH = "^[\\u4e00-\\u9fa5]+$";

/\*\*

\* Regex of username.

\* <p>scope for "a-z", "A-Z", "0-9", "\_", "Chinese character"</p>

\* <p>can't end with "\_"</p>

\* <p>length is between 6 to 20</p>

\*/

public static final String REGEX\_USERNAME = "^[\\w\\u4e00-\\u9fa5]{6,20}(?<!\_)$";

/\*\*

\* Regex of date which pattern is "yyyy-MM-dd".

\*/

public static final String REGEX\_DATE = "^(?:(?!0000)[0-9]{4}-(?:(?:0[1-9]|1[0-2])-(?:0[1-9]|1[0-9]|2[0-8])|(?:0[13-9]|1[0-2])-(?:29|30)|(?:0[13578]|1[02])-31)|(?:[0-9]{2}(?:0[48]|[2468][048]|[13579][26])|(?:0[48]|[2468][048]|[13579][26])00)-02-29)$";

/\*\*

\* Regex of ip address.

\*/

public static final String REGEX\_IP = "((2[0-4]\\d|25[0-5]|[01]?\\d\\d?)\\.){3}(2[0-4]\\d|25[0-5]|[01]?\\d\\d?)";

///////////////////////////////////////////////////////////////////////////

// The following come from http://tool.oschina.net/regex

///////////////////////////////////////////////////////////////////////////

/\*\*

\* Regex of double-byte characters.

\*/

public static final String REGEX\_DOUBLE\_BYTE\_CHAR = "[^\\x00-\\xff]";

/\*\*

\* Regex of blank line.

\*/

public static final String REGEX\_BLANK\_LINE = "\\n\\s\*\\r";

/\*\*

\* Regex of QQ number.

\*/

public static final String REGEX\_QQ\_NUM = "[1-9][0-9]{4,}";

/\*\*

\* Regex of postal code in China.

\*/

public static final String REGEX\_CHINA\_POSTAL\_CODE = "[1-9]\\d{5}(?!\\d)";

/\*\*

\* Regex of positive integer.

\*/

public static final String REGEX\_POSITIVE\_INTEGER = "^[1-9]\\d\*$";

/\*\*

\* Regex of negative integer.

\*/

public static final String REGEX\_NEGATIVE\_INTEGER = "^-[1-9]\\d\*$";

/\*\*

\* Regex of integer.

\*/

public static final String REGEX\_INTEGER = "^-?[1-9]\\d\*$";

/\*\*

\* Regex of non-negative integer.

\*/

public static final String REGEX\_NOT\_NEGATIVE\_INTEGER = "^[1-9]\\d\*|0$";

/\*\*

\* Regex of non-positive integer.

\*/

public static final String REGEX\_NOT\_POSITIVE\_INTEGER = "^-[1-9]\\d\*|0$";

/\*\*

\* Regex of positive float.

\*/

public static final String REGEX\_POSITIVE\_FLOAT = "^[1-9]\\d\*\\.\\d\*|0\\.\\d\*[1-9]\\d\*$";

/\*\*

\* Regex of negative float.

\*/

public static final String REGEX\_NEGATIVE\_FLOAT = "^-[1-9]\\d\*\\.\\d\*|-0\\.\\d\*[1-9]\\d\*$";

private RegexUtil() {

throw new UnsupportedOperationException("u can't instantiate me...");

}

///////////////////////////////////////////////////////////////////////////

// If u want more please visit http://toutiao.com/i6231678548520731137

///////////////////////////////////////////////////////////////////////////

/\*\*

\* Return whether input matches regex of simple mobile.

\*

\* @param input The input.

\* @return {@code true}: yes<br>{@code false}: no

\*/

public static boolean isMobileSimple(final CharSequence input) {

return isMatch(REGEX\_MOBILE\_SIMPLE, input);

}

/\*\*

\* Return whether input matches regex of exact mobile.

\*

\* @param input The input.

\* @return {@code true}: yes<br>{@code false}: no

\*/

public static boolean isMobileExact(final CharSequence input) {

return isMatch(REGEX\_MOBILE\_EXACT, input);

}

/\*\*

\* Return whether input matches regex of telephone number.

\*

\* @param input The input.

\* @return {@code true}: yes<br>{@code false}: no

\*/

public static boolean isTel(final CharSequence input) {

return isMatch(REGEX\_TEL, input);

}

/\*\*

\* Return whether input matches regex of id card number which length is 15.

\*

\* @param input The input.

\* @return {@code true}: yes<br>{@code false}: no

\*/

public static boolean isIDCard15(final CharSequence input) {

return isMatch(REGEX\_ID\_CARD15, input);

}

/\*\*

\* Return whether input matches regex of id card number which length is 18.

\*

\* @param input The input.

\* @return {@code true}: yes<br>{@code false}: no

\*/

public static boolean isIDCard18(final CharSequence input) {

return isMatch(REGEX\_ID\_CARD18, input);

}

/\*\*

\* Return whether input matches regex of email.

\*

\* @param input The input.

\* @return {@code true}: yes<br>{@code false}: no

\*/

public static boolean isEmail(final CharSequence input) {

return isMatch(REGEX\_EMAIL, input);

}

/\*\*

\* Return whether input matches regex of url.

\*

\* @param input The input.

\* @return {@code true}: yes<br>{@code false}: no

\*/

public static boolean isURL(final CharSequence input) {

return isMatch(REGEX\_URL, input);

}

/\*\*

\* Return whether input matches regex of Chinese character.

\*

\* @param input The input.

\* @return {@code true}: yes<br>{@code false}: no

\*/

public static boolean isZh(final CharSequence input) {

return isMatch(REGEX\_ZH, input);

}

/\*\*

\* Return whether input matches regex of username.

\* <p>scope for "a-z", "A-Z", "0-9", "\_", "Chinese character"</p>

\* <p>can't end with "\_"</p>

\* <p>length is between 6 to 20</p>.

\*

\* @param input The input.

\* @return {@code true}: yes<br>{@code false}: no

\*/

public static boolean isUsername(final CharSequence input) {

return isMatch(REGEX\_USERNAME, input);

}

/\*\*

\* Return whether input matches regex of date which pattern is "yyyy-MM-dd".

\*

\* @param input The input.

\* @return {@code true}: yes<br>{@code false}: no

\*/

public static boolean isDate(final CharSequence input) {

return isMatch(REGEX\_DATE, input);

}

/\*\*

\* Return whether input matches regex of ip address.

\*

\* @param input The input.

\* @return {@code true}: yes<br>{@code false}: no

\*/

public static boolean isIP(final CharSequence input) {

return isMatch(REGEX\_IP, input);

}

/\*\*

\* Return whether input matches the regex.

\*

\* @param regex The regex.

\* @param input The input.

\* @return {@code true}: yes<br>{@code false}: no

\*/

public static boolean isMatch(final String regex, final CharSequence input) {

return input != null && input.length() > 0 && Pattern.matches(regex, input);

}

/\*\*

\* Return the list of input matches the regex.

\*

\* @param regex The regex.

\* @param input The input.

\* @return the list of input matches the regex

\*/

public static List<String> getMatches(final String regex, final CharSequence input) {

if (input == null) return Collections.emptyList();

List<String> matches = new ArrayList<>();

Pattern pattern = Pattern.compile(regex);

Matcher matcher = pattern.matcher(input);

while (matcher.find()) {

matches.add(matcher.group());

}

return matches;

}

/\*\*

\* Splits input around matches of the regex.

\*

\* @param input The input.

\* @param regex The regex.

\* @return the array of strings computed by splitting input around matches of regex

\*/

public static String[] getSplits(final String input, final String regex) {

if (input == null) return new String[0];

return input.split(regex);

}

/\*\*

\* Replace the first subsequence of the input sequence that matches the

\* regex with the given replacement string.

\*

\* @param input The input.

\* @param regex The regex.

\* @param replacement The replacement string.

\* @return the string constructed by replacing the first matching

\* subsequence by the replacement string, substituting captured

\* subsequences as needed

\*/

public static String getReplaceFirst(final String input,

final String regex,

final String replacement) {

if (input == null) return "";

return Pattern.compile(regex).matcher(input).replaceFirst(replacement);

}

/\*\*

\* Replace every subsequence of the input sequence that matches the

\* pattern with the given replacement string.

\*

\* @param input The input.

\* @param regex The regex.

\* @param replacement The replacement string.

\* @return the string constructed by replacing each matching subsequence

\* by the replacement string, substituting captured subsequences

\* as needed

\*/

public static String getReplaceAll(final String input,

final String regex,

final String replacement) {

if (input == null) return "";

return Pattern.compile(regex).matcher(input).replaceAll(replacement);

}

}

package org.linlinjava.litemall.core.util;

import com.github.pagehelper.Page;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

/\*\*

\* 响应操作结果

\* <pre>

\* {

\* errno： 错误码，

\* errmsg：错误消息，

\* data： 响应数据

\* }

\* </pre>

\*

\* <p>

\* 错误码：

\* <ul>

\* <li> 0，成功；

\* <li> 4xx，前端错误，说明前端开发者需要重新了解后端接口使用规范：

\* <ul>

\* <li> 401，参数错误，即前端没有传递后端需要的参数；

\* <li> 402，参数值错误，即前端传递的参数值不符合后端接收范围。

\* </ul>

\* <li> 5xx，后端错误，除501外，说明后端开发者应该继续优化代码，尽量避免返回后端错误码：

\* <ul>

\* <li> 501，验证失败，即后端要求用户登录；

\* <li> 502，系统内部错误，即没有合适命名的后端内部错误；

\* <li> 503，业务不支持，即后端虽然定义了接口，但是还没有实现功能；

\* <li> 504，更新数据失效，即后端采用了乐观锁更新，而并发更新时存在数据更新失效；

\* <li> 505，更新数据失败，即后端数据库更新失败（正常情况应该更新成功）。

\* </ul>

\* <li> 6xx，小商城后端业务错误码，

\* 具体见litemall-admin-api模块的AdminResponseCode。

\* <li> 7xx，管理后台后端业务错误码，

\* 具体见litemall-wx-api模块的WxResponseCode。

\* </ul>

\*/

public class ResponseUtil {

public static Object ok() {

Map<String, Object> obj = new HashMap<String, Object>();

obj.put("errno", 0);

obj.put("errmsg", "成功");

return obj;

}

public static Object ok(Object data) {

Map<String, Object> obj = new HashMap<String, Object>();

obj.put("errno", 0);

obj.put("errmsg", "成功");

obj.put("data", data);

return obj;

}

public static Object okList(List list) {

Map<String, Object> data = new HashMap<String, Object>();

data.put("list", list);

if (list instanceof Page) {

Page page = (Page) list;

data.put("total", page.getTotal());

data.put("page", page.getPageNum());

data.put("limit", page.getPageSize());

data.put("pages", page.getPages());

}

else{

data.put("total", list.size());

data.put("page", 1);

data.put("limit", list.size());

data.put("pages", 1);

}

return ok(data);

}

public static Object okList(List list, List pagedList) {

Map<String, Object> data = new HashMap<String, Object>();

data.put("list", list);

if (pagedList instanceof Page) {

Page page = (Page) pagedList;

data.put("total", page.getTotal());

data.put("page", page.getPageNum());

data.put("limit", page.getPageSize());

data.put("pages", page.getPages());

}

else{

data.put("total", pagedList.size());

data.put("page", 1);

data.put("limit", pagedList.size());

data.put("pages", 1);

}

return ok(data);

}

public static Object fail() {

Map<String, Object> obj = new HashMap<String, Object>();

obj.put("errno", -1);

obj.put("errmsg", "错误");

return obj;

}

public static Object fail(int errno, String errmsg) {

Map<String, Object> obj = new HashMap<String, Object>();

obj.put("errno", errno);

obj.put("errmsg", errmsg);

return obj;

}

public static Object badArgument() {

return fail(401, "参数不对");

}

public static Object badArgumentValue() {

return fail(402, "参数值不对");

}

public static Object unlogin() {

return fail(501, "请登录");

}

public static Object serious() {

return fail(502, "系统内部错误");

}

public static Object unsupport() {

return fail(503, "业务不支持");

}

public static Object updatedDateExpired() {

return fail(504, "更新数据已经失效");

}

public static Object updatedDataFailed() {

return fail(505, "更新数据失败");

}

public static Object unauthz() {

return fail(506, "无操作权限");

}

}

package org.linlinjava.litemall.core.util;

import java.util.Map;

public class SystemInfoPrinter {

public static final String CREATE\_PART\_COPPER = "XOXOXOXOX";

private static int maxSize = 0;

public static void printInfo(String title, Map<String, String> infos) {

setMaxSize(infos);

printHeader(title);

for (Map.Entry<String, String> entry : infos.entrySet()) {

printLine(entry.getKey(), entry.getValue());

}

printEnd();

}

private static void setMaxSize(Map<String, String> infos) {

for (Map.Entry<String, String> entry : infos.entrySet()) {

if (entry.getValue() == null)

continue;

int size = entry.getKey().length() + entry.getValue().length();

if (size > maxSize)

maxSize = size;

}

maxSize = maxSize + 30;

}

private static void printHeader(String title) {

System.out.println(getLineCopper());

System.out.println("");

System.out.println(" " + title);

System.out.println("");

}

private static void printEnd() {

System.out.println(" ");

System.out.println(getLineCopper());

}

private static String getLineCopper() {

String copper = "";

for (int i = 0; i < maxSize; i++) {

copper += "=";

}

return copper;

}

private static void printLine(String head, String line) {

if (line == null)

return;

if (head.startsWith(CREATE\_PART\_COPPER)) {

System.out.println("");

System.out.println(" [[ " + line + " ]]");

System.out.println("");

} else {

System.out.println(" " + head + " -> " + line);

}

}

}

package org.linlinjava.litemall.core.validator;

import javax.validation.Constraint;

import javax.validation.Payload;

import java.lang.annotation.Documented;

import java.lang.annotation.Retention;

import java.lang.annotation.Target;

import static java.lang.annotation.ElementType.\*;

import static java.lang.annotation.RetentionPolicy.RUNTIME;

@Target({METHOD, FIELD, PARAMETER})

@Retention(RUNTIME)

@Documented

@Constraint(validatedBy = OrderValidator.class)

public @interface Order {

String message() default "排序类型不支持";

String[] accepts() default {"desc", "asc"};

Class<?>[] groups() default {};

Class<? extends Payload>[] payload() default {};

}

package org.linlinjava.litemall.core.validator;

import javax.validation.ConstraintValidator;

import javax.validation.ConstraintValidatorContext;

import java.util.ArrayList;

import java.util.List;

public class OrderValidator implements ConstraintValidator<Order, String> {

private List<String> valueList;

@Override

public void initialize(Order order) {

valueList = new ArrayList<String>();

for (String val : order.accepts()) {

valueList.add(val.toUpperCase());

}

}

@Override

public boolean isValid(String s, ConstraintValidatorContext constraintValidatorContext) {

if (!valueList.contains(s.toUpperCase())) {

return false;

}

return true;

}

}

package org.linlinjava.litemall.core.validator;

import javax.validation.Constraint;

import javax.validation.Payload;

import java.lang.annotation.Documented;

import java.lang.annotation.Retention;

import java.lang.annotation.Target;

import static java.lang.annotation.ElementType.\*;

import static java.lang.annotation.RetentionPolicy.RUNTIME;

@Target({METHOD, FIELD, PARAMETER})

@Retention(RUNTIME)

@Documented

@Constraint(validatedBy = SortValidator.class)

public @interface Sort {

String message() default "排序字段不支持";

String[] accepts() default {"add\_time", "id"};

Class<?>[] groups() default {};

Class<? extends Payload>[] payload() default {};

}

package org.linlinjava.litemall.core.validator;

import javax.validation.ConstraintValidator;

import javax.validation.ConstraintValidatorContext;

import java.util.ArrayList;

import java.util.List;

public class SortValidator implements ConstraintValidator<Sort, String> {

private List<String> valueList;

@Override

public void initialize(Sort sort) {

valueList = new ArrayList<String>();

for (String val : sort.accepts()) {

valueList.add(val.toUpperCase());

}

}

@Override

public boolean isValid(String s, ConstraintValidatorContext constraintValidatorContext) {

if (!valueList.contains(s.toUpperCase())) {

return false;

}

return true;

}

}

spring:

profiles:

active: core, db

messages:

encoding: UTF-8

logging:

level:

root: ERROR

org.springframework: ERROR

org.mybatis: ERROR

org.linlinjava.litemall.core: DEBUG

litemall:

# 开发者应该设置成自己的wx相关信息

wx:

app-id: wxa5b486c6b918ecfb

app-secret: e04004829d4c383b4db7769d88dfbca1

mch-id: 111111

mch-key: xxxxxx

notify-url: http://www.example.com/wx/order/pay-notify

# 商户证书文件路径

# 请参考“商户证书”一节 https://pay.weixin.qq.com/wiki/doc/api/wxa/wxa\_api.php?chapter=4\_3

key-path: xxxxx

#通知相关配置

notify:

mail:

# 邮件通知配置,邮箱一般用于接收业务通知例如收到新的订单，sendto 定义邮件接收者，通常为商城运营人员

enable: true

host: smtp.qq.com

username: 11143526@qq.com

password: evtcyhrraqyybhfi

sendfrom: 11143526@qq.com

sendto: 11143526@qq.com

# 短消息模版通知配置

# 短信息用于通知客户，例如发货短信通知，注意配置格式；template-name，template-templateId 请参考 NotifyType 枚举值

sms:

enable: false

appid: 111111111

appkey: xxxxxxxxxxxxxx

template:

- name: paySucceed

templateId: 156349

- name: captcha

templateId: 156433

- name: ship

templateId: 158002

- name: refund

templateId: 159447

# 微信模版通知配置

# 微信模版用于通知客户或者运营者，注意配置格式；template-name，template-templateId 请参考 NotifyType 枚举值

wx:

enable: false

template:

- name: paySucceed

templateId: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

- name: captcha

templateId: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

- name: ship

templateId: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

- name: refund

templateId: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

# 快鸟物流查询配置

express:

enable: false

appId: "XXXXXXXXX"

appKey: "XXXXXXXXXXXXXXXXXXXXXXXXX"

vendors:

- code: "ZTO"

name: "中通快递"

- code: "YTO"

name: "圆通速递"

- code: "YD"

name: "韵达速递"

- code: "YZPY"

name: "邮政快递包裹"

- code: "EMS"

name: "EMS"

- code: "DBL"

name: "德邦快递"

- code: "FAST"

name: "快捷快递"

- code: "ZJS"

name: "宅急送"

- code: "TNT"

name: "TNT快递"

- code: "UPS"

name: "UPS"

- code: "DHL"

name: "DHL"

- code: "FEDEX"

name: "FEDEX联邦(国内件)"

- code: "FEDEX\_GJ"

name: "FEDEX联邦(国际件)"

# 对象存储配置

storage:

# 当前工作的对象存储模式，分别是local、aliyun、tencent、qiniu

active: local

# 本地对象存储配置信息

local:

storagePath: storage

# 这个地方应该是wx模块的WxStorageController的fetch方法对应的地址

address: http://129.204.126.160:8080/wx/storage/fetch/

# 阿里云对象存储配置信息

aliyun:

endpoint: oss-cn-shenzhen.aliyuncs.com

accessKeyId: 111111

accessKeySecret: xxxxxx

bucketName: litemall

# 腾讯对象存储配置信息

# 请参考 https://cloud.tencent.com/document/product/436/6249

tencent:

secretId: 111111

secretKey: xxxxxx

region: xxxxxx

bucketName: litemall

# 七牛云对象存储配置信息

qiniu:

endpoint: http://pd5cb6ulu.bkt.clouddn.com

accessKey: 111111

secretKey: xxxxxx

bucketName: litemall