源代码文档

会懂 GEO 全域营销系统

软件全称	会懂 GEO 全域营销系统
版本号	V1.0
著作权人	四川流云流梦文化传媒有限公司
开发完成日期	2026 年 9 月 7 日
权利取得方式	原始取得
源代码规模	60 页 · 每页 50 行 · 共 3000 行

```
package com.geo.service;
import com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;
import com.baomidou.mybatisplus.core.toolkit.Wrappers;
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.geo.common.BusinessException;
import com.geo.common.PageResult;
import com.geo.entity.MonitorAlert;
import com.geo.entity.MonitorKeyword;
import com.geo.entity.OrderLog;
import com.geo.entity.WorkOrder;
import com.geo.mapper.MonitorAlertMapper;
import com.geo.mapper.MonitorKeywordMapper;
import com.geo.mapper.OrderLogMapper;
import com.geo.mapper.WorkOrderMapper;
import lombok.RequiredArgsConstructor;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Service;
import org.springframework.util.StringUtils;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
@Service
@RequiredArgsConstructor
public class MonitorService {
private final MonitorKeywordMapper keywordMapper;
private final MonitorAlertMapper alertMapper;
private final WorkOrderMapper workOrderMapper;
private final OrderLogMapper orderLogMapper;
public PageResult<MonitorKeyword> pageKeyword(String keyword, int pageNum, int pageSize) {
Page<MonitorKeyword> p = keywordMapper.selectPage(new Page<>(pageNum, pageSize),
Wrappers.<MonitorKeyword>lambdaQuery()
.like(StringUtils.hasText(keyword), MonitorKeyword::getKeyword, keyword)
.orderByDesc(MonitorKeyword::getId));
return new PageResult<>(p.getTotal(), p.getRecords());
}
public List<MonitorKeyword> listKeywords() {
return keywordMapper.selectList(Wrappers.<MonitorKeyword>lambdaQuery().eq(MonitorKeyword::getStatus,
1));
}
public void saveKeyword(MonitorKeyword k) {
if (!StringUtils.hasText(k.getKeyword())) {
throw new BusinessException("关键词必填");
}
if (keywordMapper.selectCount(Wrappers.<MonitorKeyword>lambdaQuery()
.eq(MonitorKeyword::getKeyword, k.getKeyword())) > 0) {
throw new BusinessException("关键词已存在");
}
k.setId(null);
```

```
import { ref, reactive, onMounted } from 'vue'
import { ElMessage, ElMessageBox } from 'element-plus'
import { Plus } from '@element-plus/icons-vue'
import { workflowPage, workflowSave, workflowUpdate, workflowRun, workflowDelete } from '../../api'
const flowMap = { auto_distribute: '自动分发', monitor_scan: '舆情巡检', content_check: '内容巡检',
customer_visit: '客户回访', timed_report: '定时报告' }
const flowTypeText = (t) => flowMap[t] || t || '-'
const triggerText = (t) => ({ manual: '手动', auto: '自动', cron: '定时' }[t] || t || '-')
const runText = (s) => ({ 1: '执行中', 2: '成功', 3: '失败' }[s] || '未执行')
const runTag = (s) => ({ 1: 'warning', 2: 'success', 3: 'danger' }[s] || 'info')
const list = ref([])
const total = ref(0)
const loading = ref(false)
const saving = ref(false)
const visible = ref(false)
const query = reactive({ keyword: '', pageNum: 1, pageSize: 10 })
const emptyForm = () => ({ id: null, workflowName: '', workflowType: 'auto_distribute', triggerType:
'manual', configJson: '', status: 1 })
const form = ref(emptyForm())
const resultVisible = ref(false)
const resultRow = ref(null)
async function load() {
loading.value = true
try {
const params = { pageNum: query.pageNum, pageSize: query.pageSize }
if (query.keyword) params.keyword = query.keyword
const data = await workflowPage(params)
list.value = data.records || []
total.value = data.total || 0
} finally {
loading.value = false
}
}
function openDialog(row) {
form.value = row ? { ...row } : emptyForm()
visible.value = true
}
async function save() {
if (!form.value.workflowName) {
ElMessage.warning('请填写流程名称')
return
}
if (!form.value.workflowType) {
ElMessage.warning('请选择流程类型')
return
}
saving.value = true
try {
if (form.value.id) {
await workflowUpdate(form.value)
} else {
await workflowSave(form.value)
```

```
import { ref, reactive, onMounted } from 'vue'
import { useRouter } from 'vue-router'
import { ElMessage, ElMessageBox } from 'element-plus'
import { Plus } from '@element-plus/icons-vue'
import { templatePage, templateSave, templateUpdate, templateDelete } from '../../api'
const router = useRouter()
const list = ref([])
const total = ref(0)
const loading = ref(false)
const saving = ref(false)
const visible = ref(false)
const seoVisible = ref(false)
const savingSeo = ref(false)
const query = reactive({ keyword: '', pageNum: 1, pageSize: 10 })
const emptyForm = () => ({ id: null, templateName: '', templateType: '企业官网', templatePath: '',
previewImage: '', status: 1, tdkTitle: '', tdkKeywords: '', tdkDescription: '' })
const form = ref(emptyForm())
const seoForm = ref({ id: null, tdkTitle: '', tdkKeywords: '', tdkDescription: '' })
async function load() {
loading.value = true
try {
const data = await templatePage(query)
list.value = data.records || []
total.value = data.total || 0
} finally {
loading.value = false
}
}
function openDialog(row) {
form.value = row ? { ...row } : emptyForm()
visible.value = true
}
function goDesign(row) {
router.push({ path: '/site/design', query: { templateId: row.id } })
}
function openTplSeo(row) {
seoForm.value = { id: row.id, tdkTitle: row.tdkTitle || '', tdkKeywords: row.tdkKeywords || '',
tdkDescription: row.tdkDescription || '' }
seoVisible.value = true
}
async function saveTplSeo() {
savingSeo.value = true
try {
await templateUpdate(seoForm.value)
ElMessage.success('模板 SEO 设置已保存')
seoVisible.value = false
load()
} finally {
savingSeo.value = false
}
}
async function save() {
```

```
import request from './request'
export const login = (data) => request.post('/api/auth/login', data)
export const getMe = () => request.get('/api/auth/me')
export const updatePassword = (data) => request.post('/api/auth/password', data)
export const getSummary = () => request.get('/api/dashboard/summary')
export const getTrend = () => request.get('/api/dashboard/trend')
export const getDistribution = () => request.get('/api/dashboard/distribution')
export const sitePage = (params) => request.get('/api/site/page', { params })
export const siteAll = () => request.get('/api/site/all')
export const siteSave = (data) => request.post('/api/site', data)
export const siteUpdate = (data) => request.put('/api/site', data)
export const siteDelete = (id) => request.delete('/api/site/' + id)
export const siteUpdateSeo = (data) => request.put('/api/site/seo', data)
export const siteChangeStatus = (id, status) => request.put('/api/site/' + id + '/status', null,
{ params: { status } })
export const templatePage = (params) => request.get('/api/template/page', { params })
export const templateSave = (data) => request.post('/api/template', data)
export const templateUpdate = (data) => request.put('/api/template', data)
export const templateDelete = (id) => request.delete('/api/template/' + id)
export const pageSchemaList = (params) => request.get('/api/page-schema/list', { params })
export const pageSchemaGet = (params) => request.get('/api/page-schema', { params })
export const pageSchemaUpsert = (data) => request.put('/api/page-schema', data)
export const pageSchemaDelete = (params) => request.delete('/api/page-schema', { params })
export const contentPage = (params) => request.get('/api/content/page', { params })
export const contentGet = (id) => request.get('/api/content/' + id)
export const contentSave = (data) => request.post('/api/content', data)
export const contentUpdate = (data) => request.put('/api/content', data)
export const contentDelete = (id) => request.delete('/api/content/' + id)
export const contentStatus = (id, status) => request.put('/api/content/' + id + '/status', null,
{ params: { status } })
export const platformPage = (params) => request.get('/api/platform/page', { params })
export const platformList = () => request.get('/api/platform/list')
export const platformSave = (data) => request.post('/api/platform', data)
export const platformUpdate = (data) => request.put('/api/platform', data)
export const platformDelete = (id) => request.delete('/api/platform/' + id)
export const distributeRun = (data) => request.post('/api/distribute/run', data)
export const distributePage = (params) => request.get('/api/distribute/page', { params })
export const distributeDelete = (id) => request.delete('/api/distribute/' + id)
export const distributeRetry = (id) => request.post('/api/distribute/' + id + '/retry')
export const linkPage = (params) => request.get('/api/external-link/page', { params })
export const linkSave = (data) => request.post('/api/external-link', data)
export const linkUpdate = (data) => request.put('/api/external-link', data)
export const linkDelete = (id) => request.delete('/api/external-link/' + id)
export const keywordPage = (params) => request.get('/api/monitor/keyword/page', { params })
export const keywordList = () => request.get('/api/monitor/keyword/list')
export const keywordSave = (data) => request.post('/api/monitor/keyword', data)
export const keywordUpdate = (data) => request.put('/api/monitor/keyword', data)
export const keywordDelete = (id) => request.delete('/api/monitor/keyword/' + id)
export const alertPage = (params) => request.get('/api/monitor/alert/page', { params })
export const alertHandle = (id, status, result) => request.put('/api/monitor/alert/' + id + '/handle',
null, { params: { status, result } })
export const alertToOrder = (id) => request.post('/api/monitor/alert/' + id + '/to-order')
export const riskStats = () => request.get('/api/monitor/risk-stats')
```

```
package com.geo.config;
import com.geo.entity.SysUser;
import com.geo.mapper.SysUserMapper;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.boot.CommandLineRunner;
import org.springframework.core.io.ClassPathResource;
import org.springframework.jdbc.datasource.init.ResourceDatabasePopulator;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Component;
import javax.sql.DataSource;
import java.time.LocalDateTime;
@Slf4j
@Component
@RequiredArgsConstructor
public class DataInitializer implements CommandLineRunner {
private final SysUserMapper sysUserMapper;
private final DataSource dataSource;
private final PasswordEncoder passwordEncoder;
@Override
public void run(String... args) {
try {
ResourceDatabasePopulator schema = new ResourceDatabasePopulator(new ClassPathResource("schema.sql"));
schema.setSqlScriptEncoding("UTF-8");
schema.execute(dataSource);
} catch (Exception e) {
log.warn("表结构初始化失败: {}", e.getMessage());
}
Long count = sysUserMapper.selectCount(null);
if (count != null && count > 0) {
return;
}
try {
ResourceDatabasePopulator populator = new ResourceDatabasePopulator(new ClassPathResource("data.sql"));
populator.setSqlScriptEncoding("UTF-8");
populator.execute(dataSource);
} catch (Exception e) {
log.warn("演示数据初始化跳过: {}", e.getMessage());
}
insertUser("admin", "admin123", "系统管理员", "ADMIN");
insertUser("operator", "123456", "运营专员", "USER");
log.info("默认账号初始化完成: admin/admin123, operator/123456");
}
private void insertUser(String username, String rawPassword, String realName, String role) {
if (sysUserMapper.selectCount(com.baomidou.mybatisplus.core.toolkit.Wrappers.<SysUser>lambdaQuery()
.eq(SysUser::getUsername, username)) > 0) {
return;
}
SysUser user = new SysUser();
user.setUsername(username);
```

```
import { ref, nextTick, onMounted } from 'vue'
import { ElMessage, ElMessageBox } from 'element-plus'
import { Plus } from '@element-plus/icons-vue'
import { assistantChat, assistantHistory, assistantClear, assistantSessions } from '../../api'
const SESSION_KEY = 'geo_assistant_session_id'
const WELCOME = '你好，我是 GEO 智能运营助手。可以为你解读站点数据、分发效果、舆情动态，也可以检索知识产权登记。试试问我：系统里有多少站点？'
const messages = ref([])
const sessions = ref([])
const sessionId = ref('')
const input = ref('')
const sending = ref(false)
const bodyRef = ref(null)
const modelStatus = ref(null)
async function loadModelStatus() {
try {
const d = await aiModelStatus()
modelStatus.value = d || null
} catch {
modelStatus.value = null
}
}
function genId() {
if (window.crypto && crypto.randomUUID) return crypto.randomUUID().replace(/-/g, '')
return 's' + Date.now() + Math.floor(Math.random() * 100000)
}
function ensureSession() {
if (!sessionId.value) {
sessionId.value = localStorage.getItem(SESSION_KEY) || genId()
localStorage.setItem(SESSION_KEY, sessionId.value)
}
}
function welcomeMessages() {
messages.value = [{ role: 'assistant', content: WELCOME }]
}
async function scrollBottom() {
await nextTick()
if (bodyRef.value) {
bodyRef.value.scrollTop = bodyRef.value.scrollHeight
}
}
async function loadSessions() {
const data = await assistantSessions()
sessions.value = data || []
}
async function loadHistory(sid) {
const rows = await assistantHistory(sid)
if (rows && rows.length) {
messages.value = rows.map((r) => ({ role: r.role, content: r.content }))
} else {
welcomeMessages()
```

```
import { ref, reactive, onMounted } from 'vue'
import { ElMessage, ElMessageBox } from 'element-plus'
import { Plus } from '@element-plus/icons-vue'
import { contentPage, contentSave, contentUpdate, contentDelete, contentStatus, siteAll } from
'../../api'
const list = ref([])
const sites = ref([])
const total = ref(0)
const loading = ref(false)
const saving = ref(false)
const visible = ref(false)
const typeOptions = [
{ label: '文章', value: 'article' },
{ label: '新闻', value: 'news' },
{ label: '资料', value: 'soft' },
{ label: '案例', value: 'case' }
]
const query = reactive({ siteId: null, contentType: null, keyword: '', status: null, pageNum: 1,
pageSize: 10 })
const contentMode = ref('html')
const emptyForm = () => ({ id: null, siteId: null, title: '', contentType: 'article', seoTitle: '',
seoKeywords: '', seoDescription: '', content: '', status: 0, publishTime: null })
const form = ref(emptyForm())
function typeText(v) {
const t = typeOptions.find((i) => i.value === v)
return t ? t.label : (v || '-')
}
function statusText(s) { return { 0: '草稿', 1: '待审', 2: '已发布', 3: '已下线' }[s] || '未知' }
function statusTag(s) { return { 0: 'info', 1: 'warning', 2: 'success', 3: 'danger' }[s] || 'info' }
function previewDoc() {
return form.value.content || '<div style="padding:16px;color:#909399;font-size:14px">暂无正文内容，请先
在「HTML 源码」中编辑</div>'
}
function siteName(id) {
const s = sites.value.find((i) => i.id === id)
return s ? s.siteName : '-'
}
async function load() {
loading.value = true
try {
const data = await contentPage(query)
list.value = data.records || []
total.value = data.total || 0
} finally {
loading.value = false
}
}
function openDialog(row) {
form.value = row ? { ...row } : emptyForm()
visible.value = true
}
async function save() {
if (!form.value.title || !form.value.siteId) {
ElMessage.warning('请填写标题并选择所属站点')
```

```
package com.geo.config;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.boot.ApplicationArguments;
import org.springframework.boot.ApplicationRunner;
import org.springframework.core.Ordered;
import org.springframework.core.annotation.Order;
import org.springframework.stereotype.Component;
import javax.sql.DataSource;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.sql.Timestamp;
import java.time.LocalDateTime;
import java.util.LinkedHashMap;
import java.util.Map;
@Slf4j
@Component
@RequiredArgsConstructor
@Order(Ordered.LOWEST_PRECEDENCE - 10)
public class SystemBatch4Initializer implements ApplicationRunner {
private final DataSource dataSource;
@Override
public void run(ApplicationArguments args) {
try {
ensureColumn("sys_config", "config_type", "VARCHAR(20)");
try (Connection conn = dataSource.getConnection(); Statement st = conn.createStatement()) {
st.executeUpdate("UPDATE sys_config SET config_type = 'text' WHERE config_type IS NULL OR config_type =
''");
} catch (Exception e) {
log.warn("回填配置类型失败: {}", e.getMessage());
}
int added = seedConfigRows();
log.info("SystemBatch4 引导完成，新增配置项 {} 条", added);
} catch (Exception e) {
log.warn("SystemBatch4 引导异常: {}", e.getMessage());
}
}
private void ensureColumn(String table, String column, String ddlType) {
try (Connection conn = dataSource.getConnection(); Statement st = conn.createStatement()) {
String cntSql = "SELECT COUNT(*) AS C FROM INFORMATION_SCHEMA.COLUMNS WHERE UPPER(TABLE_NAME)=UPPER('"
+ table + "') AND UPPER(COLUMN_NAME)=UPPER('" + column + "')";
boolean exists = false;
try (ResultSet rs = st.executeQuery(cntSql)) {
if (rs.next()) {
exists = rs.getLong("C") > 0;
}
}
if (exists) {
log.info("列已存在，跳过: {}.{}", table, column);
return;
```

```
import { ref, reactive, onMounted } from 'vue'
import { ElMessage, ElMessageBox } from 'element-plus'
import { Plus } from '@element-plus/icons-vue'
import { configPage, configSave, configUpdate, configDelete } from '../../api'
const typeDict = { text: '文本', number: '数字', boolean: '布尔', password: '密钥' }
const list = ref([])
const total = ref(0)
const loading = ref(false)
const saving = ref(false)
const visible = ref(false)
const query = reactive({ keyword: '', configType: '', pageNum: 1, pageSize: 10 })
const emptyForm = () => ({ id: null, configKey: '', configValue: '', configName: '', configType: 'text',
remark: '' })
const form = ref(emptyForm())
function typeTag(type) {
return type === 'password' ? 'danger' : type === 'boolean' ? 'warning' : type === 'number' ? 'info' :
'primary'
}
function isSecret(row) {
return row && row.configType === 'password'
}
async function load() {
loading.value = true
try {
const data = await configPage(query)
list.value = data.records || []
total.value = data.total || 0
} finally {
loading.value = false
}
}
function openDialog(row) {
form.value = row
? { ...row, configValue: isSecret(row) ? '' : (row.configValue ?? '') }
: emptyForm()
visible.value = true
}
async function save() {
if (!form.value.configKey) {
ElMessage.warning('请填写配置键')
return
}
saving.value = true
try {
if (form.value.id) {
const payload = { ...form.value }
if (isSecret(payload) && !payload.configValue) {
delete payload.configValue
}
await configUpdate(payload)
} else {
await configSave(form.value)
```

```
package com.geo.controller;
import com.baomidou.mybatisplus.core.toolkit.Wrappers;
import com.geo.common.Result;
import com.geo.entity.ContentInfo;
import com.geo.entity.CustomerInfo;
import com.geo.entity.MonitorAlert;
import com.geo.entity.SiteInfo;
import com.geo.entity.WorkOrder;
import com.geo.mapper.*;
import lombok.RequiredArgsConstructor;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
@RestController
@RequestMapping("/api/dashboard")
@RequiredArgsConstructor
public class DashboardController {
private final JdbcTemplate jdbcTemplate;
private final SiteInfoMapper siteInfoMapper;
private final ContentInfoMapper contentInfoMapper;
private final DistributeRecordMapper recordMapper;
private final MonitorAlertMapper alertMapper;
private final CustomerInfoMapper customerInfoMapper;
private final WorkOrderMapper workOrderMapper;
@GetMapping("/summary")
public Result<Map<String, Object>> summary() {
Map<String, Object> m = new HashMap<>();
m.put("siteCount", siteInfoMapper.selectCount(null));
m.put("contentPublished",
contentInfoMapper.selectCount(Wrappers.<ContentInfo>lambdaQuery().eq(ContentInfo::getStatus, 2)));
m.put("contentTotal", contentInfoMapper.selectCount(null));
List<com.geo.entity.DistributeRecord> records = recordMapper.selectList(null);
long expose = records.stream().mapToLong(r -> r.getExposeCount() == null ? 0 :
r.getExposeCount()).sum();
long read = records.stream().mapToLong(r -> r.getReadCount() == null ? 0 : r.getReadCount()).sum();
m.put("exposeTotal", expose);
m.put("readTotal", read);
m.put("distributeTotal", records.size());
m.put("pendingAlert",
alertMapper.selectCount(Wrappers.<MonitorAlert>lambdaQuery().eq(MonitorAlert::getAlertStatus, 0)));
m.put("alertTotal", alertMapper.selectCount(null));
m.put("customerCount", customerInfoMapper.selectCount(null));
m.put("todoOrder",
workOrderMapper.selectCount(Wrappers.<WorkOrder>lambdaQuery().lt(WorkOrder::getOrderStatus, 2)));
return Result.success(m);
}
@GetMapping("/trend")
```

```
import { ref, reactive, onMounted } from 'vue'
import { ElMessage, ElMessageBox } from 'element-plus'
import { Plus } from '@element-plus/icons-vue'
import { keywordPage, keywordSave, keywordUpdate, keywordDelete } from '../../api'
const channelMap = { email: '邮件', sms: '短信', wechat: '企微' }
const channelText = (c) => channelMap[c] || c
const list = ref([])
const total = ref(0)
const loading = ref(false)
const saving = ref(false)
const visible = ref(false)
const query = reactive({ pageNum: 1, pageSize: 10 })
const emptyForm = () => ({ id: null, keyword: '', keywordType: '品牌词', monitorLevel: 'MID', status:
1 })
const form = ref(emptyForm())
async function load() {
loading.value = true
try {
const data = await keywordPage(query)
list.value = data.records || []
total.value = data.total || 0
} finally {
loading.value = false
}
}
const levelText = (lv) => ({ HIGH: '高', MID: '中', LOW: '低' }[lv] || lv || '-')
const levelTag = (lv) => ({ HIGH: 'danger', MID: 'warning', LOW: 'info' }[lv] || 'info')
function openDialog(row) {
form.value = row ? { ...row } : emptyForm()
visible.value = true
}
async function save() {
if (!form.value.keyword) {
ElMessage.warning('请填写关键词')
return
}
saving.value = true
try {
if (form.value.id) {
await keywordUpdate(form.value)
} else {
await keywordSave(form.value)
}
ElMessage.success('保存成功')
visible.value = false
load()
} finally {
saving.value = false
}
}
function remove(row) {
```

```
package com.geo.service;
import com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;
import com.baomidou.mybatisplus.core.toolkit.Wrappers;
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.geo.common.PageResult;
import com.geo.entity.ContentInfo;
import com.geo.entity.PageSchema;
import com.geo.entity.SiteInfo;
import com.geo.entity.SiteTemplate;
import com.geo.mapper.ContentInfoMapper;
import com.geo.mapper.PageSchemaMapper;
import com.geo.mapper.SiteInfoMapper;
import com.geo.mapper.SiteTemplateMapper;
import lombok.RequiredArgsConstructor;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Service;
import org.springframework.util.StringUtils;
import java.time.LocalDateTime;
import java.util.Collections;
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
@Service
@RequiredArgsConstructor
public class SiteService {
private final SiteInfoMapper siteInfoMapper;
private final SiteTemplateMapper siteTemplateMapper;
private final ContentInfoMapper contentInfoMapper;
private final PageSchemaMapper pageSchemaMapper;
private final ObjectMapper objectMapper;
private String currentUser() {
try {
return SecurityContextHolder.getContext().getAuthentication().getName();
} catch (Exception e) {
return "system";
}
}
public PageResult<SiteInfo> pageSite(String keyword, Integer status, int pageNum, int pageSize) {
LambdaQueryWrapper<SiteInfo> q = Wrappers.<SiteInfo>lambdaQuery()
.like(StringUtils.hasText(keyword), SiteInfo::getSiteName, keyword)
.eq(status != null, SiteInfo::getStatus, status)
.orderByDesc(SiteInfo::getId);
Page<SiteInfo> p = siteInfoMapper.selectPage(new Page<>(pageNum, pageSize), q);
return new PageResult<>(p.getTotal(), p.getRecords());
}
public List<SiteInfo> listAllSites() {
return siteInfoMapper.selectList(Wrappers.<SiteInfo>lambdaQuery().orderByDesc(SiteInfo::getId));
}
```

```
package com.geo.controller;
import com.geo.common.PageResult;
import com.geo.common.Result;
import com.geo.entity.MonitorKeyword;
import com.geo.service.MonitorService;
import lombok.RequiredArgsConstructor;
import org.springframework.web.bind.annotation.*;
import java.util.List;
import java.util.Map;
@RestController
@RequestMapping("/api/monitor")
@RequiredArgsConstructor
public class MonitorController {
private final MonitorService monitorService;
@GetMapping("/keyword/page")
public Result<PageResult<MonitorKeyword>> pageKeyword(@RequestParam(required = false) String keyword,
@RequestParam(defaultValue = "1") int pageNum,
@RequestParam(defaultValue = "10") int pageSize) {
return Result.success(monitorService.pageKeyword(keyword, pageNum, pageSize));
}
@GetMapping("/keyword/list")
public Result<List<MonitorKeyword>> listKeyword() {
return Result.success(monitorService.listKeywords());
}
@PostMapping("/keyword")
public Result<Void> saveKeyword(@RequestBody MonitorKeyword k) {
monitorService.saveKeyword(k);
return Result.success();
}
@PutMapping("/keyword")
public Result<Void> updateKeyword(@RequestBody MonitorKeyword k) {
monitorService.updateKeyword(k);
return Result.success();
}
@DeleteMapping("/keyword/{id}")
public Result<Void> deleteKeyword(@PathVariable Long id) {
monitorService.deleteKeyword(id);
return Result.success();
}
@GetMapping("/alert/page")
public Result<PageResult<Map<String, Object>>> pageAlert(@RequestParam(required = false) Long keywordId,
@RequestParam(required = false) String level,
@RequestParam(required = false) Integer status,
@RequestParam(defaultValue = "1") int pageNum,
@RequestParam(defaultValue = "10") int pageSize) {
return Result.success(monitorService.pageAlert(keywordId, level, status, pageNum, pageSize));
}
@PutMapping("/alert/{id}/handle")
public Result<Void> handle(@PathVariable Long id, @RequestParam Integer status,
@RequestParam(required = false) String result) {
```

```
import { ref, reactive, onMounted } from 'vue'
import { ElMessage, ElMessageBox } from 'element-plus'
import { Delete } from '@element-plus/icons-vue'
import { logPage, logDetail, logClean } from '../../api'
const moduleOptions = [
'认证登录', '系统管理', '站点管理', '模板管理', '内容管理', '全域分发', '分发平台',
'外链资源', '舆情监测', '客户服务', 'AI 能力中心', '页面设计', '文件管理', '站点预览', '数据看板'
]
const list = ref([])
const total = ref(0)
const loading = ref(false)
const cleaning = ref(false)
const detailVisible = ref(false)
const cleanVisible = ref(false)
const cleanMode = ref('keepDays')
const cleanRange = ref([])
const cleanKeepDays = ref(90)
const detail = ref(null)
const range = ref([])
const query = reactive({ operator: '', module: '', operation: '', status: null, pageNum: 1, pageSize:
10 })
async function load() {
loading.value = true
try {
const params = { ...query, status: query.status ?? undefined }
if (range.value && range.value.length === 2) {
params.startTime = range.value[0]
params.endTime = range.value[1]
}
const data = await logPage(params)
list.value = data.records || []
total.value = data.total || 0
} finally {
loading.value = false
}
}
function methodName(method) {
if (!method) return '-'
const m = method.split(' ')[0]
return m || '-'
}
function methodType(method) {
const m = (method || '').split(' ')[0]
if (m === 'DELETE') return 'danger'
if (m === 'POST') return 'warning'
if (m === 'PUT') return 'primary'
return 'info'
}
async function openDetail(row) {
detail.value = await logDetail(row.id)
detailVisible.value = true
```

```
package com.geo.config;
import com.baomidou.mybatisplus.core.toolkit.Wrappers;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.geo.entity.ContentInfo;
import com.geo.entity.PageSchema;
import com.geo.entity.SiteInfo;
import com.geo.entity.SiteTemplate;
import com.geo.mapper.ContentInfoMapper;
import com.geo.mapper.PageSchemaMapper;
import com.geo.mapper.SiteInfoMapper;
import com.geo.mapper.SiteTemplateMapper;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.boot.ApplicationArguments;
import org.springframework.boot.ApplicationRunner;
import org.springframework.core.Ordered;
import org.springframework.core.annotation.Order;
import org.springframework.stereotype.Component;
import org.springframework.util.StringUtils;
import javax.sql.DataSource;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import java.sql.Timestamp;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Set;
@Slf4j
@Component
@RequiredArgsConstructor
@Order(Ordered.LOWEST_PRECEDENCE)
public class BootstrapRunner implements ApplicationRunner {
private static final String SITE4_NAME = "华东制造网";
private static final String CITY_SITE_NAME = "成都生活服务分站";
private static final String CITY_DOMAIN = "chengdu.example.com";
private static final String BIZ_SITE_NAME = "川渝明珠商务咨询官网";
private static final String BIZ_DOMAIN = "chuanyu-business.example.com";
private static final Set<String> REGISTERED_BLOCK_KEYS = Set.of(
"hero", "textHtml", "features", "products", "services", "cases", "newsList",
"statsBand", "askPanel", "qaList", "hotSidebar", "cityBar", "cityHot",
"cityWeather", "cityServices", "cityNotices", "relatedList", "twoCol");
private final DataSource dataSource;
private final SiteInfoMapper siteInfoMapper;
private final ContentInfoMapper contentInfoMapper;
private final SiteTemplateMapper siteTemplateMapper;
private final PageSchemaMapper pageSchemaMapper;
```

```
package com.geo.service;
import com.baomidou.mybatisplus.core.toolkit.Wrappers;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.node.ArrayNode;
import com.fasterxml.jackson.databind.node.ObjectNode;
import com.geo.common.BusinessException;
import com.geo.entity.SysConfig;
import com.geo.mapper.SysConfigMapper;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.stereotype.Service;
import org.springframework.util.StringUtils;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.time.Duration;
import java.util.LinkedHashMap;
import java.util.Map;
@Slf4j
@Service
@RequiredArgsConstructor
public class AiLlmClient {
public static final String CFG_ENABLED = "sys.ai.enabled";
public static final String CFG_PROVIDER = "sys.ai.provider";
public static final String CFG_BASE_URL = "sys.ai.baseUrl";
public static final String CFG_API_KEY = "sys.ai.apiKey";
public static final String CFG_MODEL = "sys.ai.model";
public static final String CFG_TIMEOUT = "sys.ai.timeoutSecs";
private static final Map<String, String[]> PROVIDER_DEFAULTS = Map.of(
"deepseek", new String[]{"https://api.deepseek.com/v1", "deepseek-chat"},
"openai", new String[]{"https://api.openai.com/v1", "gpt-4o-mini"},
"qwen", new String[]{"https://dashscope.aliyuncs.com/compatible-mode/v1", "qwen-plus"}
);
private final SysConfigMapper configMapper;
private final ObjectMapper objectMapper;
private final HttpClient httpClient = HttpClient.newBuilder()
.connectTimeout(Duration.ofSeconds(10))
.version(HttpClient.Version.HTTP_1_1)
.build();
private String cfg(String key) {
SysConfig c = configMapper.selectOne(Wrappers.<SysConfig>lambdaQuery()
.eq(SysConfig::getConfigKey, key)
.last("LIMIT 1"));
return c == null ? null : c.getConfigValue();
}
private boolean enabled() {
String v = cfg(CFG_ENABLED);
return v == null || "true".equalsIgnoreCase(v.trim());
```

```
package com.geo.service;
import com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;
import com.baomidou.mybatisplus.core.toolkit.Wrappers;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.geo.common.BusinessException;
import com.geo.entity.PageSchema;
import com.geo.mapper.PageSchemaMapper;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;
import org.springframework.util.StringUtils;
import java.time.LocalDateTime;
import java.util.List;
import java.util.Map;
@Service
@RequiredArgsConstructor
public class PageSchemaService {
private final PageSchemaMapper pageSchemaMapper;
private final ObjectMapper objectMapper;
public List<PageSchema> list(Long templateId, Long siteId) {
return pageSchemaMapper.selectList(Wrappers.<PageSchema>lambdaQuery()
.eq(templateId != null, PageSchema::getTemplateId, templateId)
.eq(siteId != null, PageSchema::getSiteId, siteId)
.orderByDesc(PageSchema::getUpdateTime));
}
public PageSchema getOne(Long templateId, Long siteId, String pageKey) {
return pageSchemaMapper.selectOne(findWrapper(templateId, siteId, pageKey));
}
public void upsert(Map<String, Object> body) {
Long templateId = body.get("templateId") == null ? null : ((Number) body.get("templateId")).longValue();
Long siteId = body.get("siteId") == null ? null : ((Number) body.get("siteId")).longValue();
String pageKey = body.get("pageKey") == null ? null : String.valueOf(body.get("pageKey"));
if (!StringUtils.hasText(pageKey)) {
throw new BusinessException("pageKey 必填");
}
if (templateId == null && siteId == null) {
throw new BusinessException("templateId 与 siteId 至少传一个");
}
String blocksJson = toBlocksJson(body.get("blocks"));
String seoJson = toSeoJson(body.get("seo"), body.get("seoJson"));
LocalDateTime now = LocalDateTime.now();
PageSchema exist = pageSchemaMapper.selectOne(findWrapper(templateId, siteId, pageKey));
if (exist == null) {
PageSchema row = new PageSchema();
row.setTemplateId(templateId);
row.setSiteId(siteId);
row.setPageKey(pageKey);
row.setBlocks(blocksJson);
row.setSeoJson(seoJson);
row.setStatus(1);
row.setCreateTime(now);
```

```
import { ref, onMounted, nextTick } from 'vue'
import { useRouter } from 'vue-router'
import * as echarts from 'echarts'
import { getSummary, getTrend, getDistribution } from '../api'
const router = useRouter()
const cards = ref([])
const trendRef = ref(null)
const distRef = ref(null)
let trendChart = null
let distChart = null
function go(path) {
router.push(path)
}
async function loadSummary() {
const d = await getSummary()
cards.value = [
{ label: '托管站点', value: d.siteCount, icon: 'Monitor', color: '#409eff', bg: '#ecf5ff', path:
'/site/list' },
{ label: '已发布内容', value: d.contentPublished, icon: 'Document', color: '#67c23a', bg: '#f0f9eb',
path: '/site/content' },
{ label: '全域曝光', value: d.exposeTotal, icon: 'View', color: '#e6a23c', bg: '#fdf6ec', path:
'/distribute/record' },
{ label: '内容阅读', value: d.readTotal, icon: 'Reading', color: '#909399', bg: '#f4f4f5', path:
'/site/content' },
{ label: '待处理预警', value: d.pendingAlert, icon: 'AlarmClock', color: '#f56c6c', bg: '#fef0f0', path:
'/monitor/alert' },
{ label: '进行中工单', value: d.todoOrder, icon: 'Tickets', color: '#9c27b0', bg: '#f3e5f5', path:
'/crm/order' }
]
}
async function loadTrend() {
const d = await getTrend()
if (!trendChart) {
trendChart = echarts.init(trendRef.value)
}
trendChart.setOption({
tooltip: { trigger: 'axis' },
legend: { data: ['内容发布', '分发任务', '曝光量'] },
grid: { left: 40, right: 20, top: 40, bottom: 30 },
xAxis: { type: 'category', data: d.dates },
yAxis: { type: 'value' },
series: [
{ name: '内容发布', type: 'line', smooth: true, data: d.content, areaStyle: { opacity: 0.1 }, itemStyle:
{ color: '#409eff' } },
{ name: '分发任务', type: 'line', smooth: true, data: d.distribute, areaStyle: { opacity: 0.1 },
itemStyle: { color: '#67c23a' } },
{ name: '曝光量', type: 'bar', barWidth: 12, data: d.expose, itemStyle: { color: '#e6a23c' } }
]
})
}
async function loadDistribution() {
const list = await getDistribution()
if (!distChart) {
distChart = echarts.init(distRef.value)
}
distChart.setOption({
tooltip: { trigger: 'item', formatter: '{b}: {c} 次 ({d}%)' },
```

```
legend: { bottom: 0, type: 'scroll' },
```

```
package com.geo.service;
import com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;
import com.baomidou.mybatisplus.core.toolkit.Wrappers;
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.geo.common.BusinessException;
import com.geo.common.PageResult;
import com.geo.entity.OperationLog;
import com.geo.entity.SysConfig;
import com.geo.entity.SysUser;
import com.geo.mapper.OperationLogMapper;
import com.geo.mapper.SysConfigMapper;
import com.geo.mapper.SysUserMapper;
import lombok.RequiredArgsConstructor;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;
import org.springframework.util.StringUtils;
import java.time.LocalDateTime;
import java.util.List;
@Service
@RequiredArgsConstructor
public class SystemService {
private final SysUserMapper userMapper;
private final OperationLogMapper logMapper;
private final SysConfigMapper configMapper;
private final PasswordEncoder passwordEncoder;
public PageResult<SysUser> pageUser(String keyword, String role, Integer status, int pageNum, int
pageSize) {
Page<SysUser> p = userMapper.selectPage(new Page<>(pageNum, pageSize),
Wrappers.<SysUser>lambdaQuery()
.and(StringUtils.hasText(keyword), w -> w
.like(SysUser::getUsername, keyword)
.or().like(SysUser::getRealName, keyword))
.eq(StringUtils.hasText(role), SysUser::getRole, role)
.eq(status != null, SysUser::getStatus, status)
.orderByDesc(SysUser::getId));
for (SysUser u : p.getRecords()) {
u.setPassword(null);
}
return new PageResult<>(p.getTotal(), p.getRecords());
}
public void saveUser(SysUser user) {
if (!StringUtils.hasText(user.getUsername())) {
throw new BusinessException("用户名必填");
}
if (userMapper.selectCount(Wrappers.<SysUser>lambdaQuery().eq(SysUser::getUsername,
user.getUsername())) > 0) {
throw new BusinessException("用户名已存在");
}
user.setId(null);
user.setPassword(passwordEncoder.encode(StringUtils.hasText(user.getPassword()) ? user.getPassword() :
"123456"));
user.setCreateTime(LocalDateTime.now());
user.setUpdateTime(LocalDateTime.now());
```

```
import { ref, reactive, onMounted } from 'vue'
import { ElMessage, ElMessageBox } from 'element-plus'
import { Plus } from '@element-plus/icons-vue'
import { platformPage, platformSave, platformUpdate, platformDelete } from '../../api'
const typeMap = { search: '搜索引擎', social: '社交媒体', video: '短视频', qa: '问答社区', news: '新闻媒体
', wiki: '百科平台' }
const typeText = (t) => typeMap[t] || t || '-'
const list = ref([])
const total = ref(0)
const loading = ref(false)
const saving = ref(false)
const visible = ref(false)
const query = reactive({ keyword: '', pageNum: 1, pageSize: 10 })
const emptyForm = () => ({ id: null, platformName: '', platformCode: '', platformAccount: '',
accountToken: '', accountConfig: '', status: 1 })
const form = ref(emptyForm())
async function load() {
loading.value = true
try {
const data = await platformPage(query)
list.value = data.records || []
total.value = data.total || 0
} finally {
loading.value = false
}
}
function openDialog(row) {
form.value = row ? { ...row } : emptyForm()
visible.value = true
}
async function save() {
if (!form.value.platformName) {
ElMessage.warning('请填写平台名称')
return
}
saving.value = true
try {
if (form.value.id) {
await platformUpdate(form.value)
} else {
await platformSave(form.value)
}
ElMessage.success('保存成功')
visible.value = false
load()
} finally {
saving.value = false
}
}
function remove(row) {
ElMessageBox.confirm(`确认删除平台「${row.platformName}」？`, '删除确认', { type: 'warning' }).then(async
() => {
await platformDelete(row.id)
```

```
{ path: '/preview/template/:id/list/:type', name: 'tpl-list', component: () =>
import('../views/site/SitePreview.vue'), meta: { public: true } },
{ path: '/preview/template/:id/detail/:cid', name: 'tpl-detail', component: () =>
import('../views/site/SitePreview.vue'), meta: { public: true } },
{ path: '/preview/template/:id/page/:page', name: 'tpl-page', component: () =>
import('../views/site/SitePreview.vue'), meta: { public: true } },
{
path: '/',
component: Layout,
redirect: '/dashboard',
children: [
{ path: 'dashboard', component: () => import('../views/dashboard.vue'), meta: { title: '数据看板', icon:
'DataBoard' } },
{ path: 'site/design', component: () => import('../views/site/DesignView.vue'), meta: { title: '页面设
计', public: false } },
{
path: 'site',
redirect: '/site/list',
meta: { title: '站点与内容', icon: 'Monitor' },
children: [
{ path: 'list', component: () => import('../views/site/SiteList.vue'), meta: { title: '站点管理' } },
{ path: 'template', component: () => import('../views/site/TemplateList.vue'), meta: { title: '模板管理
' } },
{ path: 'content', component: () => import('../views/site/ContentList.vue'), meta: { title: '内容管理
' } }
]
},
{
path: 'distribute',
redirect: '/distribute/record',
meta: { title: '全域分发', icon: 'Promotion' },
children: [
{ path: 'record', component: () => import('../views/distribute/RecordList.vue'), meta: { title: '分发任
务' } },
{ path: 'platform', component: () => import('../views/distribute/PlatformList.vue'), meta: { title: '分
发平台' } },
{ path: 'link', component: () => import('../views/distribute/LinkList.vue'), meta: { title: '外链资源
' } }
]
},
{
path: 'monitor',
redirect: '/monitor/alert',
meta: { title: '舆情监测', icon: 'AlarmClock' },
children: [
{ path: 'alert', component: () => import('../views/monitor/AlertList.vue'), meta: { title: '舆情预警
' } },
{ path: 'keyword', component: () => import('../views/monitor/KeywordList.vue'), meta: { title: '监测关
键词' } }
]
},
{
path: 'crm',
redirect: '/crm/customer',
meta: { title: '客户服务', icon: 'UserFilled' },
children: [
{ path: 'customer', component: () => import('../views/crm/CustomerList.vue'), meta: { title: '客户管理
```

```
' } },
{ path: 'order', component: () => import('../views/crm/OrderList.vue'), meta: { title: '工单中心' } }
]
},
{
path: 'ai',
```

```
package com.geo.controller;
import com.geo.common.PageResult;
import com.geo.common.Result;
import com.geo.entity.AiReport;
import com.geo.entity.AssistantMessage;
import com.geo.entity.KnowledgeProperty;
import com.geo.entity.WorkflowConfig;
import com.geo.service.AiService;
import lombok.RequiredArgsConstructor;
import org.springframework.web.bind.annotation.*;
import java.util.List;
import java.util.Map;
@RestController
@RequestMapping("/api/ai")
@RequiredArgsConstructor
public class AiController {
private final AiService aiService;
@GetMapping("/report/page")
public Result<PageResult<AiReport>> pageReport(@RequestParam(required = false) String type,
@RequestParam(defaultValue = "1") int pageNum,
@RequestParam(defaultValue = "10") int pageSize) {
return Result.success(aiService.pageReport(type, pageNum, pageSize));
}
@GetMapping("/report/{id}")
public Result<AiReport> getReport(@PathVariable Long id) {
return Result.success(aiService.getReport(id));
}
@PostMapping("/report/generate")
public Result<AiReport> generate(@RequestParam(required = false) String reportType,
@RequestParam(required = false) String period,
@RequestParam(required = false) String reportTitle) {
return Result.success(aiService.generateReport(reportType, period, reportTitle));
}
@DeleteMapping("/report/{id}")
public Result<Void> deleteReport(@PathVariable Long id) {
aiService.deleteReport(id);
return Result.success();
}
@GetMapping("/workflow/page")
public Result<PageResult<WorkflowConfig>> pageWorkflow(@RequestParam(required = false) String keyword,
@RequestParam(defaultValue = "1") int pageNum,
@RequestParam(defaultValue = "10") int pageSize) {
return Result.success(aiService.pageWorkflow(keyword, pageNum, pageSize));
}
@PostMapping("/workflow")
public Result<Void> saveWorkflow(@RequestBody WorkflowConfig w) {
aiService.saveWorkflow(w);
return Result.success();
}
@PutMapping("/workflow")
```

```
const MARK = 'data-geo-seo'
const LD_ID = 'geo-jsonld'
function clean() {
document.querySelectorAll(`[${MARK}]`).forEach((el) => el.remove())
const ld = document.getElementById(LD_ID)
if (ld) ld.remove()
}
function addMeta(attrKey, attrVal, content) {
const m = document.createElement('meta')
m.setAttribute(MARK, '1')
m.setAttribute(attrKey, attrVal)
m.setAttribute('content', content || '')
document.head.appendChild(m)
}
function addLink(rel, href) {
if (!href) return
const l = document.createElement('link')
l.setAttribute(MARK, '1')
l.rel = rel
l.href = href
document.head.appendChild(l)
}
function addJsonLd(obj) {
if (!obj) return
const s = document.createElement('script')
s.type = 'application/ld+json'
s.id = LD_ID
s.textContent = JSON.stringify(obj)
document.head.appendChild(s)
}
export function applySeo(opts = {}) {
clean()
const {
title = '',
description = '',
keywords = '',
canonical = '',
robots = 'index,follow',
ogType = 'website',
ogImage = '',
jsonLd = null
} = opts
if (title) document.title = title
if (description) addMeta('name', 'description', description)
if (keywords) addMeta('name', 'keywords', keywords)
addMeta('name', 'robots', robots)
addMeta('property', 'og:title', title)
addMeta('property', 'og:type', ogType)
addMeta('property', 'og:url', canonical || location.href)
if (description) addMeta('property', 'og:description', description)
```

```
package com.geo.config;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.boot.ApplicationArguments;
import org.springframework.boot.ApplicationRunner;
import org.springframework.core.Ordered;
import org.springframework.core.annotation.Order;
import org.springframework.stereotype.Component;
import javax.sql.DataSource;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Timestamp;
import java.time.LocalDateTime;
import java.util.LinkedHashMap;
import java.util.Map;
@Slf4j
@Component
@RequiredArgsConstructor
@Order(Ordered.LOWEST_PRECEDENCE - 20)
public class AiConfigInitializer implements ApplicationRunner {
private final DataSource dataSource;
@Override
public void run(ApplicationArguments args) {
int added = 0;
Map<String, String[]> rows = new LinkedHashMap<>();
rows.put("sys.ai.provider", new String[]{"deepseek", "AI 厂商", "text", "大模型厂商: deepseek / openai /
qwen / custom(custom 需显式填写 baseUrl)"});
rows.put("sys.ai.baseUrl", new String[]{"https://api.deepseek.com/v1", "AI 接口地址", "text", "OpenAI 兼
容接口地址(以 /v1 结尾); 留空则按厂商默认"});
rows.put("sys.ai.model", new String[]{"deepseek-chat", "AI 模型名", "text", "模型名, 如 deepseek-chat /
qwen-plus / gpt-4o-mini; 留空则按厂商默认"});
rows.put("sys.ai.timeoutSecs", new String[]{"60", "AI 调用超时(秒)", "number", "大模型接口调用超时时间, 默
认 60 秒"});
try (Connection conn = dataSource.getConnection()) {
for (Map.Entry<String, String[]> e : rows.entrySet()) {
String key = e.getKey();
String[] v = e.getValue();
long cnt;
try (PreparedStatement ps = conn.prepareStatement("SELECT COUNT(*) AS C FROM sys_config WHERE
config_key = ?")) {
ps.setString(1, key);
try (ResultSet rs = ps.executeQuery()) {
rs.next();
cnt = rs.getLong("C");
}
}
if (cnt > 0) {
continue;
}
Timestamp now = Timestamp.valueOf(LocalDateTime.now());
try (PreparedStatement ps = conn.prepareStatement("INSERT INTO sys_config (config_key, config_value,
config_name, config_type, remark, create_time, update_time) VALUES (?, ?, ?, ?, ?, ?, ?)")) {
ps.setString(1, key);
ps.setString(2, v[0]);
ps.setString(3, v[1]);
```

```
package com.geo.controller;
import com.geo.common.PageResult;
import com.geo.common.Result;
import com.geo.entity.SiteInfo;
import com.geo.service.SiteService;
import lombok.RequiredArgsConstructor;
import org.springframework.web.bind.annotation.*;
import java.util.List;
import java.util.Map;
@RestController
@RequestMapping("/api/site")
@RequiredArgsConstructor
public class SiteController {
private final SiteService siteService;
@GetMapping("/page")
public Result<PageResult<SiteInfo>> page(@RequestParam(required = false) String keyword,
@RequestParam(required = false) Integer status,
@RequestParam(defaultValue = "1") int pageNum,
@RequestParam(defaultValue = "10") int pageSize) {
return Result.success(siteService.pageSite(keyword, status, pageNum, pageSize));
}
@GetMapping("/all")
public Result<List<SiteInfo>> all() {
return Result.success(siteService.listAllSites());
}
@GetMapping("/{id}")
public Result<SiteInfo> get(@PathVariable Long id) {
return Result.success(siteService.getSite(id));
}
@PostMapping
public Result<Void> save(@RequestBody SiteInfo site) {
siteService.saveSite(site);
return Result.success();
}
@PutMapping
public Result<Void> update(@RequestBody SiteInfo site) {
siteService.updateSite(site);
return Result.success();
}
@DeleteMapping("/{id}")
public Result<Void> delete(@PathVariable Long id) {
siteService.deleteSite(id);
return Result.success();
}
@PutMapping("/{id}/status")
public Result<Void> changeStatus(@PathVariable Long id, @RequestParam int status) {
siteService.changeSiteStatus(id, status);
return Result.success();
}
@PutMapping("/seo")
```

```
import { computed } from 'vue'
import { resolveSkin } from './skin.js'
import BlockRenderer from './blocks/BlockRenderer.vue'
const props = defineProps({
site: { type: Object, required: true },
tpl: { type: Object, default: null },
contents: { type: Array, default: () => [] },
type: { type: String, default: 'all' },
pageConfig: { type: Object, default: () => ({}) }
})
const skinKey = computed(() => resolveSkin(props.tpl, props.site))
const isQa = computed(() => skinKey.value === 'qa')
const isPortal = computed(() => skinKey.value === 'portal')
const isCity = computed(() => skinKey.value === 'city')
const listCfg = computed(() => (props.pageConfig && props.pageConfig.list &&
Array.isArray(props.pageConfig.list.blocks)) ? props.pageConfig.list.blocks : null)
const listBlocks = computed(() => (listCfg.value || []).filter((b) => b && b.blockKey && b.enabled !==
false))
const blockCtx = computed(() => ({ site: props.site, tpl: props.tpl, contents: props.contents, skinKey:
skinKey.value, pageKey: 'list', type: props.type }))
const TYPE_NAMES = {
all: '全部内容',
news: '新闻动态',
article: '深度文章',
case: '客户案例',
product: '产品服务',
soft: '资料下载'
}
const columns = [
{ key: 'all', name: '全部内容' },
{ key: 'news', name: '新闻动态' },
{ key: 'article', name: '深度文章' },
{ key: 'case', name: '客户案例' },
{ key: 'product', name: '产品服务' },
{ key: 'soft', name: '资料下载' }
]
const QA_TYPE_NAMES = {
all: '全部问答',
news: '问答动态',
article: '问答广场',
case: '案例讨论',
product: '装修方案',
soft: '装修知识库'
}
const CITY_TYPE_NAMES = {
all: '本地信息',
news: '本地资讯',
article: '城市话题',
case: '民生专题',
product: '本地服务',
soft: '便民资料'
}
const portalColumns = [
```

```
export const TYPE_META = {
all: { label: '全部内容', icon: 'fa-solid fa-layer-group' },
news: { label: '新闻动态', icon: 'fa-solid fa-newspaper' },
article: { label: '深度文章', icon: 'fa-solid fa-file-lines' },
case: { label: '客户案例', icon: 'fa-solid fa-briefcase' },
product: { label: '产品服务', icon: 'fa-solid fa-cube' },
soft: { label: '资料下载', icon: 'fa-solid fa-download' }
}
export const SKIN_INFO = {
'corp-standard': {
label: '云帆官网标准版',
tagline: '官网标准版 · 科技蓝调',
brandIcon: 'fa-solid fa-sailboat',
heroIcon: 'fa-solid fa-rocket',
desc: '现代企业官网，科技蓝调精致商务，产品 / 案例 / 新闻全栏目优雅呈现。'
},
'corp-business': {
label: '川渝明珠商务版',
tagline: '官网商务版 · 金棕大气',
brandIcon: 'fa-solid fa-crown',
heroIcon: 'fa-solid fa-gem',
desc: '高端商务官网，金棕配色大气稳重，服务 / 案例展示企业实力。'
},
portal: {
label: '绿色能源门户版',
tagline: '资讯门户版 · 清新自然',
brandIcon: 'fa-solid fa-leaf',
heroIcon: 'fa-solid fa-sun',
desc: '新能源资讯门户，绿色清新资讯流，政策 / 技术侧栏丰富专业。'
},
qa: {
label: '装修问答版',
tagline: '问答社区版 · 暖色互助',
brandIcon: 'fa-solid fa-home',
heroIcon: 'fa-solid fa-question-circle',
desc: '装修问答社区，暖色温馨，提问 / 回答 / 达人话题轻松互动。'
},
city: {
label: '城市分站版',
tagline: '城市门户版 · 蓝色城市',
brandIcon: 'fa-solid fa-city',
heroIcon: 'fa-solid fa-map-location-dot',
desc: '城市本地门户，蓝色城市界面，天气 / 便民 / 本地热点一站式聚合。'
}
}
export const SKIN_ORDER = ['corp-standard', 'corp-business', 'portal', 'qa', 'city']
export function resolveSkin(tpl = null, site = null) {
const type = (tpl && tpl.templateType) || (site && site.siteType) || ''
const name = (tpl && tpl.templateName) || ''
const siteName = (site && site.siteName) || ''
```

```
import { ref, reactive, onMounted } from 'vue'
import { ElMessage, ElMessageBox } from 'element-plus'
import { Plus } from '@element-plus/icons-vue'
import { knowledgePage, knowledgeGet, knowledgeSave, knowledgeUpdate, knowledgeDelete } from
'../../api'
const typeOptions = ['专利', '商标', '软著', '版权', '美术作品']
const typeColorMap = { 专利: 'danger', 商标: 'primary', 软著: 'warning', 版权: 'success', 美术作品:
'info' }
const typeText = (t) => typeOptions.includes(t) ? t : (t || '-')
const typeTag = (t) => typeColorMap[t] || 'info'
const list = ref([])
const total = ref(0)
const loading = ref(false)
const saving = ref(false)
const visible = ref(false)
const detailVisible = ref(false)
const detail = ref(null)
const query = reactive({ keyword: '', type: '', status: null, pageNum: 1, pageSize: 10 })
const emptyForm = () => ({ id: null, propertyName: '', propertyNo: '', propertyType: '商标', holder: '',
status: 1, applyDate: null, registerDate: null, expireDate: null, remark: '' })
const form = ref(emptyForm())
async function load() {
loading.value = true
try {
const params = { pageNum: query.pageNum, pageSize: query.pageSize }
if (query.keyword) params.keyword = query.keyword
if (query.type) params.type = query.type
if (query.status !== null && query.status !== '') params.status = query.status
const data = await knowledgePage(params)
list.value = data.records || []
total.value = data.total || 0
} finally {
loading.value = false
}
}
function openDialog(row) {
form.value = row ? { ...row } : emptyForm()
visible.value = true
}
async function save() {
if (!form.value.propertyName) {
ElMessage.warning('请填写产权名称')
return
}
if (!form.value.propertyType) {
ElMessage.warning('请选择产权类型')
return
}
saving.value = true
try {
if (form.value.id) {
await knowledgeUpdate(form.value)
} else {
```

```
export const DEFAULT_STRUCTURE = {
'corp-standard': {
home: [
{
blockKey: 'hero', enabled: true,
props: {
tag: 'SMART DIGITAL · 企业数字化服务商',
title: '{{site.tdkTitle}}',
sub: '{{site.tdkDescription}}',
primaryText: '开始体验',
primaryHref: '/list/product',
secondaryText: '了解更多',
secondaryHref: '/page/about',
items: []
}
},
{
blockKey: 'features', enabled: true,
props: {
kicker: 'Core Capabilities', title: '核心能力', sub: '以技术创新为引擎，赋能千行百业',
items: [
{ icon: 'fa-solid fa-bolt', title: '极速部署', desc: '容器化架构，分钟级完成环境搭建，大幅缩短上线周期。' },
{ icon: 'fa-solid fa-shield-halved', title: '安全可靠', desc: '多层次安全防护体系，数据加密与合规审计，保障
业务稳定。' },
{ icon: 'fa-solid fa-chart-line', title: '智能分析', desc: 'AI 驱动的数据洞察，实时业务看板，辅助精准决策。
' }
]
}
},
{
blockKey: 'products', enabled: true,
props: { kicker: 'Solutions', title: '产品与服务', sub: '面向企业的一站式数字化解决方案', contentRef:
'product', max: 4 }
},
{
blockKey: 'textHtml', enabled: true,
props: {
kicker: 'About', title: '让数字转型更简单', html:
'<p>{{site.tdkDescription || "欢迎访问 {{site.siteName}}，本站持续更新专业、可靠、可读的内容。"}}</p>' +
'<ul class="geo-about-points"><li><i class="fa-solid fa-circle-check"></i><span>依托会懂 GEO 全域营销系
统，统一管理内容资产与品牌阵地</span></li>' +
'<li><i class="fa-solid fa-arrows-rotate"></i><span>定期同步行业政策、产品与技术进展，保证信息鲜活可靠
</span></li>' +
'<li><i class="fa-solid fa-headset"></i><span>专业编辑团队持续供稿，7×24 在线支持站点运营
</span></li></ul>',
align: 'left'
}
},
{
blockKey: 'newsList', enabled: true,
props: { kicker: 'News', title: '新闻动态', contentRef: 'news', max: 4, cols: 1 }
}
],
list: [
{
blockKey: 'hotSidebar', enabled: true,
```

```
import { ref, computed } from 'vue'
import { useRoute, useRouter } from 'vue-router'
import { ElMessage, ElMessageBox } from 'element-plus'
import { useUserStore } from '../store/user'
import { updatePassword } from '../api'
const store = useUserStore()
const route = useRoute()
const router = useRouter()
const menus = [
{ path: '/dashboard', title: '数据看板', icon: 'DataBoard' },
{
path: '/site', title: '站点与内容', icon: 'Monitor',
children: [
{ path: '/site/list', title: '站点管理' },
{ path: '/site/template', title: '模板管理' },
{ path: '/site/content', title: '内容管理' }
]
},
{
path: '/distribute', title: '全域分发', icon: 'Promotion',
children: [
{ path: '/distribute/record', title: '分发任务' },
{ path: '/distribute/platform', title: '分发平台' },
{ path: '/distribute/link', title: '外链资源' }
]
},
{
path: '/monitor', title: '舆情监测', icon: 'AlarmClock',
children: [
{ path: '/monitor/alert', title: '舆情预警' },
{ path: '/monitor/keyword', title: '监测关键词' }
]
},
{
path: '/crm', title: '客户服务', icon: 'UserFilled',
children: [
{ path: '/crm/customer', title: '客户管理' },
{ path: '/crm/order', title: '工单中心' }
]
},
{
path: '/ai', title: 'AI 能力中心', icon: 'MagicStick',
children: [
{ path: '/ai/overview', title: 'AI 总览' },
{ path: '/ai/report', title: 'AI 报告' },
{ path: '/ai/workflow', title: '工作流' },
{ path: '/ai/assistant', title: '智能助手' },
{ path: '/ai/knowledge', title: '知识产权' }
]
},
```

```
package com.geo.aspect;
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.node.ObjectNode;
import com.geo.entity.OperationLog;
import com.geo.mapper.OperationLogMapper;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.aspectj.lang.ProceedingJoinPoint;
import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Aspect;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Component;
import org.springframework.util.StringUtils;
import org.springframework.validation.BindingResult;
import org.springframework.web.context.request.RequestContextHolder;
import org.springframework.web.context.request.ServletRequestAttributes;
import org.springframework.web.multipart.MultipartFile;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
@Slf4j
@Aspect
@Component
@RequiredArgsConstructor
public class OperationLogAspect {
private static final int PARAMS_MAX = 2000;
private static final int ERROR_MAX = 400;
private static final Pattern USERNAME_PATTERN = Pattern.compile("\"username\"\\s*:\\s*\"([^\"]+)\"");
private final OperationLogMapper logMapper;
private final ObjectMapper objectMapper;
@Around("@within(org.springframework.web.bind.annotation.RestController)")
public Object around(ProceedingJoinPoint pjp) throws Throwable {
ServletRequestAttributes attrs = (ServletRequestAttributes) RequestContextHolder.getRequestAttributes();
if (attrs == null) {
return pjp.proceed();
}
HttpServletRequest request = attrs.getRequest();
String httpMethod = request.getMethod();
String uri = request.getRequestURI();
if (uri == null || !uri.startsWith("/api/")) {
return pjp.proceed();
}
```

```
import { ref, reactive, onMounted } from 'vue'
import { ElMessage, ElMessageBox } from 'element-plus'
import { alertPage, alertHandle, alertToOrder, riskStats, keywordList } from '../../api'
const list = ref([])
const total = ref(0)
const loading = ref(false)
const keywords = ref([])
const risk = ref({})
const handleVisible = ref(false)
const currentRow = ref(null)
const handleResult = ref('')
const query = reactive({ level: null, status: null, pageNum: 1, pageSize: 10 })
const levelText = (lv) => ({ HIGH: '高', MID: '中', LOW: '低' }[lv] || lv || '-')
const levelTag = (lv) => ({ HIGH: 'danger', MID: 'warning', LOW: 'info' }[lv] || 'info')
const statusText = (s) => ({ 0: '未处理', 1: '处理中', 2: '已处理', 3: '已转工单' }[s] || '未知')
const statusTag = (s) => ({ 0: 'danger', 1: 'warning', 2: 'success', 3: 'info' }[s] || 'info')
function keywordName(id) {
const k = keywords.value.find((i) => i.id === id)
return k ? k.keyword : 'ID:' + id
}
async function load() {
loading.value = true
try {
const data = await alertPage(query)
list.value = data.records || []
total.value = data.total || 0
} finally {
loading.value = false
}
}
async function loadRisk() {
risk.value = await riskStats()
}
function toOrder(row) {
ElMessageBox.confirm('确认将该预警转成客户工单？', '转工单', { type: 'warning' }).then(async () => {
await alertToOrder(row.id)
ElMessage.success('已转为工单')
load()
loadRisk()
}).catch(() => {})
}
function beginHandle(row) {
ElMessageBox.confirm('开始处理该预警？状态将流转为「处理中」。', '开始处理', { type: 'info' }).then(async ()
=> {
await alertHandle(row.id, 1, '开始处理')
ElMessage.success('已开始处理')
load()
loadRisk()
}).catch(() => {})
}
function handle(row) {
```

```
import { ref, reactive, onMounted } from 'vue'
import { useRouter } from 'vue-router'
import { ElMessage, ElMessageBox } from 'element-plus'
import { Plus } from '@element-plus/icons-vue'
import { sitePage, siteSave, siteUpdate, siteDelete, siteUpdateSeo, siteChangeStatus, templatePage }
from '../../api'
const router = useRouter()
const list = ref([])
const total = ref(0)
const loading = ref(false)
const saving = ref(false)
const visible = ref(false)
const seoVisible = ref(false)
const query = reactive({ keyword: '', status: null, pageNum: 1, pageSize: 10 })
const emptyForm = () => ({ id: null, siteName: '', domain: '', siteType: '企业官网', templateId: null,
status: 1, tdkTitle: '', tdkKeywords: '', tdkDescription: '' })
const form = ref(emptyForm())
const seoForm = ref({ id: null, tdkTitle: '', tdkKeywords: '', tdkDescription: '' })
const templates = ref([])
const templateMap = ref({})
async function load() {
loading.value = true
try {
const data = await sitePage(query)
list.value = data.records || []
total.value = data.total || 0
list.value.forEach(r => { r.templateName = templateMap.value[r.templateId] || '-' })
} finally {
loading.value = false
}
}
async function loadTemplates() {
const data = await templatePage({ pageNum: 1, pageSize: 200 })
templates.value = data.records || []
const m = {}
templates.value.forEach(t => { m[t.id] = t.templateName })
templateMap.value = m
}
async function toggleStatus(row, status) {
const action = status === 1 ? '启用' : '停用'
try {
await siteChangeStatus(row.id, status)
ElMessage.success(`站点已${action}`)
} catch (e) {
row.status = status === 1 ? 0 : 1
ElMessage.error(`${action}失败，已恢复原状态`)
}
}
function openDialog(row) {
form.value = row ? { ...row } : emptyForm()
visible.value = true
}
```

```
package com.geo.controller;
import com.geo.common.PageResult;
import com.geo.common.Result;
import com.geo.entity.CustomerInfo;
import com.geo.entity.ServiceRecord;
import com.geo.entity.WorkOrder;
import com.geo.service.CrmService;
import lombok.RequiredArgsConstructor;
import org.springframework.web.bind.annotation.*;
import java.util.Map;
@RestController
@RequestMapping("/api/crm")
@RequiredArgsConstructor
public class CrmController {
private final CrmService crmService;
@GetMapping("/customer/page")
public Result<PageResult<CustomerInfo>> pageCustomer(@RequestParam(required = false) String keyword,
@RequestParam(required = false) String source,
@RequestParam(required = false) String createBy,
@RequestParam(required = false) Integer status,
@RequestParam(defaultValue = "1") int pageNum,
@RequestParam(defaultValue = "10") int pageSize) {
return Result.success(crmService.pageCustomer(keyword, source, createBy, status, pageNum, pageSize));
}
@PostMapping("/customer")
public Result<Void> saveCustomer(@RequestBody CustomerInfo c) {
crmService.saveCustomer(c);
return Result.success();
}
@PutMapping("/customer")
public Result<Void> updateCustomer(@RequestBody CustomerInfo c) {
crmService.updateCustomer(c);
return Result.success();
}
@DeleteMapping("/customer/{id}")
public Result<Void> deleteCustomer(@PathVariable Long id) {
crmService.deleteCustomer(id);
return Result.success();
}
@GetMapping("/customer/{id}/detail")
public Result<Map<String, Object>> customerDetail(@PathVariable Long id) {
return Result.success(crmService.customerDetail(id));
}
@PutMapping("/customer/{id}/status")
public Result<Void> changeCustomerStatus(@PathVariable Long id, @RequestParam Integer status) {
crmService.changeCustomerStatus(id, status);
return Result.success();
}
@GetMapping("/order/page")
public Result<PageResult<Map<String, Object>>> pageOrder(@RequestParam(required = false) String keyword,
```

```
package com.geo.controller;
import com.geo.common.PageResult;
import com.geo.common.Result;
import com.geo.entity.OperationLog;
import com.geo.entity.SysConfig;
import com.geo.entity.SysUser;
import com.geo.service.SystemService;
import lombok.RequiredArgsConstructor;
import org.springframework.web.bind.annotation.*;
import java.util.List;
import java.util.Map;
@RestController
@RequestMapping("/api/system")
@RequiredArgsConstructor
public class SystemController {
private final SystemService systemService;
@GetMapping("/user/page")
public Result<PageResult<SysUser>> pageUser(@RequestParam(required = false) String keyword,
@RequestParam(required = false) String role,
@RequestParam(required = false) Integer status,
@RequestParam(defaultValue = "1") int pageNum,
@RequestParam(defaultValue = "10") int pageSize) {
return Result.success(systemService.pageUser(keyword, role, status, pageNum, pageSize));
}
@PostMapping("/user")
public Result<Void> saveUser(@RequestBody SysUser user) {
systemService.saveUser(user);
return Result.success();
}
@PutMapping("/user")
public Result<Void> updateUser(@RequestBody SysUser user) {
systemService.updateUser(user);
return Result.success();
}
@DeleteMapping("/user/{id}")
public Result<Void> deleteUser(@PathVariable Long id) {
systemService.deleteUser(id);
return Result.success();
}
@PutMapping("/user/{id}/reset-password")
public Result<Void> resetPassword(@PathVariable Long id, @RequestParam(required = false) String
password) {
systemService.resetPassword(id, password);
return Result.success();
}
@GetMapping("/log/page")
public Result<PageResult<OperationLog>> pageLog(@RequestParam(required = false) String module,
@RequestParam(required = false) String operator,
@RequestParam(required = false) String operation,
@RequestParam(required = false) Integer status,
@RequestParam(required = false) String startTime,
```

```
package com.geo.service;
import com.baomidou.mybatisplus.core.toolkit.Wrappers;
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.geo.common.BusinessException;
import com.geo.common.PageResult;
import com.geo.entity.AiReport;
import com.geo.entity.AssistantMessage;
import com.geo.entity.ContentInfo;
import com.geo.entity.DistributeRecord;
import com.geo.entity.KnowledgeProperty;
import com.geo.entity.MonitorAlert;
import com.geo.entity.SiteInfo;
import com.geo.entity.WorkflowConfig;
import com.geo.mapper.AiReportMapper;
import com.geo.mapper.AssistantMessageMapper;
import com.geo.mapper.ContentInfoMapper;
import com.geo.mapper.CustomerInfoMapper;
import com.geo.mapper.DistributeRecordMapper;
import com.geo.mapper.KnowledgePropertyMapper;
import com.geo.mapper.MonitorAlertMapper;
import com.geo.mapper.MonitorKeywordMapper;
import com.geo.mapper.ServiceRecordMapper;
import com.geo.mapper.SiteInfoMapper;
import com.geo.mapper.WorkOrderMapper;
import com.geo.mapper.WorkflowConfigMapper;
import jakarta.annotation.PreDestroy;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Service;
import org.springframework.util.StringUtils;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
import java.util.UUID;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.stream.Collectors;
@Slf4j
@Service
@RequiredArgsConstructor
public class AiService {
private final AiReportMapper reportMapper;
private final WorkflowConfigMapper workflowMapper;
private final KnowledgePropertyMapper knowledgeMapper;
```

```
package com.geo.common;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.nio.charset.StandardCharsets;
import java.util.Base64;
import java.util.Date;
@Component
public class JwtUtil {
@Value("${geo.jwt.secret}")
private String secret;
@Value("${geo.jwt.expire-hours:24}")
private long expireHours;
private static final Base64.Encoder B64 = Base64.getUrlEncoder().withoutPadding();
private static final Base64.Decoder B64D = Base64.getUrlDecoder();
public String generateToken(String username, String role) {
String header = b64("{\"alg\":\"HS256\",\"typ\":\"JWT\"}");
long now = System.currentTimeMillis();
long exp = now + expireHours * 3600_000L;
String payload = b64("{\"sub\":\"" + username + "\",\"role\":\"" + role
+ "\",\"iat\":" + (now / 1000) + ",\"exp\":" + (exp / 1000) + "}");
String signingInput = header + "." + payload;
return signingInput + "." + hmac(signingInput);
}
public String getUsername(String token) {
return parseClaim(token, "sub");
}
public String getRole(String token) {
return parseClaim(token, "role");
}
public boolean validate(String token) {
try {
String[] parts = token.split("\\.");
if (parts.length != 3) {
return false;
}
String signingInput = parts[0] + "." + parts[1];
String expected = hmac(signingInput);
if (!expected.equals(parts[2])) {
return false;
}
String payloadJson = new String(B64D.decode(parts[1]), StandardCharsets.UTF_8);
long exp = extractLong(payloadJson, "exp");
return exp > System.currentTimeMillis() / 1000;
} catch (Exception e) {
return false;
}
}
private String parseClaim(String token, String key) {
```

```
import { ref, reactive, onMounted } from 'vue'
import { ElMessage, ElMessageBox } from 'element-plus'
import { Plus } from '@element-plus/icons-vue'
import { orderPage, orderSave, orderUpdate, orderFinish, orderAccept, orderAssign, orderClose,
orderDelete, orderDetail, customerPage, userPage } from '../../api'
const statusMap = { 0: '待处理', 1: '处理中', 2: '已完成', 3: '已关闭' }
const statusText = (v) => statusMap[v] ?? '未知'
const statusType = (v) => ['danger', 'warning', 'success', 'info'][v] || 'info'
const priorityMap = { HIGH: '高', MID: '中', LOW: '低' }
const priorityText = (v) => priorityMap[v] || v || '-'
const priorityType = (v) => ({ HIGH: 'danger', MID: 'warning', LOW: 'info' })[v] || 'info'
const orderTypeMap = { consult: '咨询建议', technical: '技术故障', change: '需求变更', aftersale: '售后服
务', alert: '舆情处置' }
const orderTypeText = (v) => orderTypeMap[v] || v || '-'
const actionType = (a) => ({ create: 'primary', accept: 'success', assign: 'warning', complete:
'success', close: 'danger', update: 'info', process: 'warning' })[a] || 'info'
const list = ref([])
const customers = ref([])
const users = ref([])
const total = ref(0)
const loading = ref(false)
const saving = ref(false)
const savingAssign = ref(false)
const visible = ref(false)
const detailVisible = ref(false)
const detail = ref(null)
const assignVisible = ref(false)
const assignForm = ref({ orderId: null, assignee: null, remark: '' })
const query = reactive({ keyword: '', customerId: null, status: null, priority: null, orderType: null,
source: null, pageNum: 1, pageSize: 10 })
const emptyForm = () => ({
id: null, customerId: null, orderDesc: '', orderType: 'consult', priority: 'MID',
orderStatus: 0, assignee: null, attachFiles: '', handleResult: '', orderSource: 'manual'
})
const form = ref(emptyForm())
function customerName(id) {
const c = customers.value.find((i) => i.id === id)
return c ? c.customerName : 'ID:' + id
}
async function load() {
loading.value = true
try {
const data = await orderPage(query)
list.value = data.records || []
total.value = data.total || 0
} finally {
loading.value = false
}
}
function resetQuery() {
query.keyword = ''
query.customerId = null
query.status = null
query.priority = null
```

```
package com.geo.service;
import com.baomidou.mybatisplus.core.toolkit.Wrappers;
import com.geo.common.BusinessException;
import com.geo.common.JwtUtil;
import com.geo.entity.SysUser;
import com.geo.mapper.SysUserMapper;
import lombok.RequiredArgsConstructor;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;
import java.util.HashMap;
import java.util.Map;
@Service
@RequiredArgsConstructor
public class AuthService {
private final SysUserMapper userMapper;
private final JwtUtil jwtUtil;
private final PasswordEncoder passwordEncoder;
public Map<String, Object> login(String username, String password) {
SysUser user = userMapper.selectOne(Wrappers.<SysUser>lambdaQuery().eq(SysUser::getUsername, username));
if (user == null || user.getPassword() == null
|| !passwordEncoder.matches(password == null ? "" : password, user.getPassword())) {
throw new BusinessException("用户名或密码错误");
}
if (user.getStatus() == null || user.getStatus() != 1) {
throw new BusinessException("账号已被禁用");
}
Map<String, Object> result = new HashMap<>();
result.put("token", jwtUtil.generateToken(user.getUsername(), user.getRole()));
result.put("user", toSafe(user));
return result;
}
public SysUser currentUser() {
String username = SecurityContextHolder.getContext().getAuthentication().getName();
return userMapper.selectOne(Wrappers.<SysUser>lambdaQuery().eq(SysUser::getUsername, username));
}
public Map<String, Object> me() {
return toSafe(currentUser());
}
public void updatePassword(String oldPassword, String newPassword) {
SysUser user = currentUser();
if (!passwordEncoder.matches(oldPassword == null ? "" : oldPassword, user.getPassword())) {
throw new BusinessException("原密码错误");
}
SysUser update = new SysUser();
update.setId(user.getId());
update.setPassword(passwordEncoder.encode(newPassword));
userMapper.updateById(update);
}
public static Map<String, Object> toSafe(SysUser user) {
```

```
import { ref, reactive, onMounted, onBeforeUnmount } from 'vue'
import { useRoute } from 'vue-router'
import { ElMessage, ElMessageBox } from 'element-plus'
import { Plus } from '@element-plus/icons-vue'
import { reportPage, reportGet, reportGenerate, reportDelete } from '../../api'
const route = useRoute()
const typeMap = { seo: 'SEO 诊断', risk: '舆情风险', operate: '分发运营' }
const typeText = (t) => typeMap[t] || t || '-'
const typeTag = (t) => ({ seo: 'primary', risk: 'danger', operate: 'warning' }[t] || 'info')
const statusText = (s) => ({ 1: '生成中', 2: '成功', 3: '失败' }[s] || '-')
const statusTag = (s) => ({ 1: 'warning', 2: 'success', 3: 'danger' }[s] || 'info')
const periodText = (p) => ({ week: '周报', month: '月报' }[p] || p || '-')
const list = ref([])
const total = ref(0)
const loading = ref(false)
const query = reactive({ type: '', period: '', pageNum: 1, pageSize: 10 })
const genVisible = ref(false)
const genLoading = ref(false)
const genError = ref('')
const emptyGen = () => ({ reportType: 'seo', period: 'month', reportTitle: '' })
const genForm = ref(emptyGen())
const detailVisible = ref(false)
const detail = ref(null)
let pollTimer = null
async function load() {
loading.value = true
try {
const params = { pageNum: query.pageNum, pageSize: query.pageSize }
if (query.type) params.type = query.type
if (query.period) params.period = query.period
const data = await reportPage(params)
list.value = data.records || []
total.value = data.total || 0
schedulePoll()
} finally {
loading.value = false
}
}
function schedulePoll() {
clearTimeout(pollTimer)
const pending = (list.value || []).some((r) => r.status === 1)
if (pending) {
pollTimer = setTimeout(() => load(), 3000)
}
}
function openGenerate(row) {
genError.value = ''
if (row) {
genForm.value = { reportType: row.reportType || 'seo', period: row.reportPeriod || 'month', reportTitle:
'' }
} else {
```

```
import { ref, reactive, onMounted } from 'vue'
import { ElMessage, ElMessageBox } from 'element-plus'
import { Plus } from '@element-plus/icons-vue'
import { linkPage, linkSave, linkUpdate, linkDelete } from '../../api'
const list = ref([])
const total = ref(0)
const loading = ref(false)
const saving = ref(false)
const visible = ref(false)
const query = reactive({ keyword: '', pageNum: 1, pageSize: 10 })
const emptyForm = () => ({ id: null, title: '', linkUrl: '', linkType: '0', sourceSite: '', status: 1 })
const form = ref(emptyForm())
async function load() {
loading.value = true
try {
const data = await linkPage(query)
list.value = data.records || []
total.value = data.total || 0
} finally {
loading.value = false
}
}
function openDialog(row) {
form.value = row ? { ...row } : emptyForm()
visible.value = true
}
async function save() {
if (!form.value.title || !form.value.linkUrl) {
ElMessage.warning('请填写名称和地址')
return
}
saving.value = true
try {
if (form.value.id) {
await linkUpdate(form.value)
} else {
await linkSave(form.value)
}
ElMessage.success('保存成功')
visible.value = false
load()
} finally {
saving.value = false
}
}
function remove(row) {
ElMessageBox.confirm(`确认删除「${row.title}」？`, '删除确认', { type: 'warning' }).then(async () => {
await linkDelete(row.id)
ElMessage.success('已删除')
load()
```

```
import { ref, reactive, computed, onMounted } from 'vue'
import { ElMessage, ElMessageBox } from 'element-plus'
import { Promotion } from '@element-plus/icons-vue'
import { distributePage, distributeRun, distributeRetry, distributeDelete, contentPage, platformList }
from '../../api'
const STATUS_META = [
{ value: 0, label: '待分发', tag: 'info' },
{ value: 1, label: '分发中', tag: 'warning' },
{ value: 2, label: '成功', tag: 'success' },
{ value: 3, label: '失败', tag: 'danger' }
]
const statusText = (v) => (STATUS_META.find((s) => s.value === v) || {}).label || '未知'
const statusTag = (v) => (STATUS_META.find((s) => s.value === v) || {}).tag || 'info'
const list = ref([])
const total = ref(0)
const loading = ref(false)
const running = ref(false)
const dialogVisible = ref(false)
const contents = ref([])
const platforms = ref([])
const query = reactive({ contentId: null, platformId: null, status: null, pageNum: 1, pageSize: 10 })
const distForm = ref({ contentId: null, platformIds: [] })
const publishedContents = computed(() => contents.value.filter((c) => c.status === 2))
const enabledPlatforms = computed(() => platforms.value.filter((p) => p.status === 1))
function contentTitle(id) {
const c = contents.value.find((i) => i.id === id)
return c ? c.title : 'ID:' + id
}
function platformName(id) {
const p = platforms.value.find((i) => i.id === id)
return p ? p.platformName : 'ID:' + id
}
function resolveUrl(url) {
if (!url) return ''
return /^https?:\/\
}
async function load() {
loading.value = true
try {
const data = await distributePage(query)
list.value = data.records || []
total.value = data.total || 0
} finally {
loading.value = false
}
}
function openDist() {
distForm.value = { contentId: null, platformIds: [] }
dialogVisible.value = true
}
async function run() {
```

```
return set.slice(0, 3)
}
if (skinKey.value === 'corp-business') {
const set = []
if (productItems.value.length) set.push({ label: '解决方案', value: productItems.value.length + ' 项',
icon: 'fa-solid fa-briefcase' })
if (caseItems.value.length) set.push({ label: '经典案例', value: caseItems.value.length + ' 个', icon:
'fa-solid fa-award' })
if (newsItems.value.length) set.push({ label: '行业洞察', value: newsItems.value.length + ' 篇', icon:
'fa-solid fa-chart-line' })
return set
}
return []
})
const corpFeatures = [
{ icon: 'fa-solid fa-bolt', title: '极速部署', desc: '容器化架构，分钟级完成环境搭建，大幅缩短上线周期。' },
{ icon: 'fa-solid fa-shield-halved', title: '安全可靠', desc: '多层次安全防护体系，数据加密与合规审计，保障
业务稳定。' },
{ icon: 'fa-solid fa-chart-line', title: '智能分析', desc: 'AI 驱动的数据洞察，实时业务看板，辅助精准决策。
' }
]
const corpPoints = [
{ icon: 'fa-solid fa-circle-check', text: '依托会懂 GEO 全域营销系统，统一管理内容资产与品牌阵地' },
{ icon: 'fa-solid fa-arrows-rotate', text: '定期同步行业政策、产品与技术进展，保证信息鲜活可靠' },
{ icon: 'fa-solid fa-headset', text: '专业编辑团队持续供稿，7×24 在线支持站点运营' }
]
const serviceItems = computed(() => {
const fallback = [
{ icon: 'fa-solid fa-lightbulb', title: '战略咨询', desc: '市场洞察、商业模式设计、战略规划，为企业提供顶层
智慧支持。' },
{ icon: 'fa-solid fa-bullhorn', title: '品牌营销', desc: '品牌定位、整合营销传播、数字营销，提升品牌影响力和
市场占有率。' },
{ icon: 'fa-solid fa-hand-holding-dollar', title: '资本运作', desc: '投融资顾问、并购重组、上市辅导，助力企
业资本价值最大化。' }
]
const items = productItems.value.map((c) => ({ icon: 'fa-solid fa-gem', title: c.title, desc:
excerpt(c), link: c.id }))
return items.length >= 2 ? items.slice(0, 3) : fallback
})
const portalTags = [
{ icon: 'fa-solid fa-bolt', label: '光伏' },
{ icon: 'fa-solid fa-wind', label: '风电' },
{ icon: 'fa-solid fa-battery-full', label: '储能' },
{ icon: 'fa-solid fa-car', label: '新能源车' }
]
const sideFalls = ['《能源法》配套细则征求意见', '碳排放权交易管理办法修订', '分布式光伏发电补贴政策延续']
const techFalls = ['钙钛矿电池转换效率突破 32%', '海上风电漂浮式技术新进展', '氢能储运成本降低 20%']
function sideItems(t, n) {
const arr = t === 'article' ? articleItems.value : t === 'news' ? newsItems.value : softItems.value
return arr.slice(0, n)
}
const portalItems = computed(() => (newsItems.value.length ? newsItems.value : articleItems.value))
const portalRest = computed(() => portalItems.value.slice(1, 6))
const portalTech = computed(() => {
const soft = sideItems('soft', 3)
if (soft.length) return soft.map((c) => ({ id: c.id, title: c.title }))
const news = sideItems('news', 3).slice(1)
```

```
if (news.length) return news.map((c) => ({ id: c.id, title: c.title }))
return techFalls.map((t) => ({ id: null, title: t }))
```

```
import { ref, reactive, onMounted } from 'vue'
import { ElMessage, ElMessageBox } from 'element-plus'
import { Plus } from '@element-plus/icons-vue'
import { customerPage, customerSave, customerUpdate, customerDelete, customerDetail,
customerChangeStatus, serviceSave, serviceDelete, userPage } from '../../api'
const statusMap = { 0: '潜在', 1: '跟进中', 2: '已成交', 3: '流失' }
const statusText = (v) => statusMap[v] ?? '未知'
const levelMap = {
KEY: { name: 'A·重要', tag: 'danger' },
IMPORTANT: { name: 'B·重点', tag: 'warning' },
NORMAL: { name: 'C·普通', tag: 'primary' },
COMMON: { name: 'D·一般', tag: 'info' }
}
const levelOptions = Object.keys(levelMap).map((k) => ({ value: k, name: levelMap[k].name }))
const levelOf = (v) => levelMap[v]
const levelName = (v) => (levelMap[v] ? levelMap[v].name : '-')
const sourceOptions = ['site', 'content', 'monitor', 'referral', 'exhibition', '其他']
const serviceTypeOptions = ['咨询服务', '实施服务', '培训服务', '内容服务', '售后服务', '回访跟进']
const orderTypeMap = { consult: '咨询建议', technical: '技术故障', change: '需求变更', aftersale: '售后服
务', alert: '舆情处置' }
const orderTypeText = (v) => orderTypeMap[v] || v || '-'
const orderStatusMap = { 0: '待处理', 1: '处理中', 2: '已完成', 3: '已关闭' }
const orderStatusType = (v) => ['danger', 'warning', 'success', 'info'][v] || 'info'
const orderStatusText = (v) => orderStatusMap[v] ?? '未知'
const tagOptions = ['高意向', '待首访', '官网咨询', '内容获客', '舆情触达', '已签约', '重点客户', '新能源',
'装修行业']
const list = ref([])
const total = ref(0)
const loading = ref(false)
const saving = ref(false)
const visible = ref(false)
const detailVisible = ref(false)
const detail = ref(null)
const users = ref([])
const savingSvc = ref(false)
const query = reactive({ keyword: '', source: null, status: null, createBy: null, pageNum: 1, pageSize:
10 })
const emptyForm = () => ({
id: null, customerName: '', company: '', source: '', contactPerson: '', phone: '',
email: '', customerLevel: 'NORMAL', status: 0, tags: '', remark: '', tagList: []
})
const form = ref(emptyForm())
const emptySvc = (customerId) => ({ customerId, serviceType: '', serviceTime: null, content: '', result:
'', nextFollowTime: null })
const svcForm = ref(emptySvc(null))
const tagList = (tags) => (tags ? String(tags).split(',').map((t) => t.trim()).filter(Boolean) : [])
async function load() {
loading.value = true
try {
const data = await customerPage(query)
list.value = data.records || []
total.value = data.total || 0
} finally {
loading.value = false
}
```

```
package com.geo.service;
import com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;
import com.baomidou.mybatisplus.core.toolkit.Wrappers;
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.geo.common.BusinessException;
import com.geo.common.PageResult;
import com.geo.entity.ContentInfo;
import com.geo.entity.DistributePlatform;
import com.geo.entity.DistributeRecord;
import com.geo.entity.ExternalLink;
import com.geo.entity.SiteInfo;
import com.geo.entity.SiteTemplate;
import com.geo.mapper.ContentInfoMapper;
import com.geo.mapper.DistributePlatformMapper;
import com.geo.mapper.DistributeRecordMapper;
import com.geo.mapper.ExternalLinkMapper;
import com.geo.mapper.SiteInfoMapper;
import com.geo.mapper.SiteTemplateMapper;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;
import org.springframework.util.StringUtils;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
@Service
@RequiredArgsConstructor
public class DistributeService {
private final DistributePlatformMapper platformMapper;
private final DistributeRecordMapper recordMapper;
private final ContentInfoMapper contentInfoMapper;
private final ExternalLinkMapper externalLinkMapper;
private final SiteInfoMapper siteInfoMapper;
private final SiteTemplateMapper siteTemplateMapper;
public static final int STATUS_WAIT = 0;
public static final int STATUS_RUNNING = 1;
public static final int STATUS_SUCCESS = 2;
public static final int STATUS_FAIL = 3;
public PageResult<DistributePlatform> pagePlatform(String keyword, int pageNum, int pageSize) {
Page<DistributePlatform> p = platformMapper.selectPage(new Page<>(pageNum, pageSize),
Wrappers.<DistributePlatform>lambdaQuery()
.like(StringUtils.hasText(keyword), DistributePlatform::getPlatformName, keyword)
.orderByDesc(DistributePlatform::getId));
return new PageResult<>(p.getTotal(), p.getRecords());
}
public List<DistributePlatform> listPlatforms() {
return platformMapper.selectList(Wrappers.<DistributePlatform>lambdaQuery()
.eq(DistributePlatform::getStatus, 1)
.orderByAsc(DistributePlatform::getId));
```

```
package com.geo.service;
import com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;
import com.baomidou.mybatisplus.core.toolkit.Wrappers;
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.geo.common.BusinessException;
import com.geo.common.PageResult;
import com.geo.entity.CustomerInfo;
import com.geo.entity.OrderLog;
import com.geo.entity.ServiceRecord;
import com.geo.entity.SysUser;
import com.geo.entity.WorkOrder;
import com.geo.mapper.CustomerInfoMapper;
import com.geo.mapper.OrderLogMapper;
import com.geo.mapper.ServiceRecordMapper;
import com.geo.mapper.SysUserMapper;
import com.geo.mapper.WorkOrderMapper;
import lombok.RequiredArgsConstructor;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Service;
import org.springframework.util.StringUtils;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
@Service
@RequiredArgsConstructor
public class CrmService {
private final CustomerInfoMapper customerMapper;
private final WorkOrderMapper workOrderMapper;
private final ServiceRecordMapper serviceRecordMapper;
private final OrderLogMapper orderLogMapper;
private final SysUserMapper sysUserMapper;
private String currentUser() {
try {
return SecurityContextHolder.getContext().getAuthentication().getName();
} catch (Exception e) {
return "system";
}
}
public PageResult<CustomerInfo> pageCustomer(String keyword, String source, String createBy,
Integer status, int pageNum, int pageSize) {
LambdaQueryWrapper<CustomerInfo> q = Wrappers.<CustomerInfo>lambdaQuery()
.and(StringUtils.hasText(keyword), w -> w.like(CustomerInfo::getCustomerName, keyword)
.or().like(CustomerInfo::getCompany, keyword)
.or().like(CustomerInfo::getPhone, keyword)
.or().like(CustomerInfo::getContactPerson, keyword))
.eq(StringUtils.hasText(source), CustomerInfo::getSource, source)
```

```
export const TYPE_NAMES = {
all: '全部内容', news: '新闻动态', article: '深度文章', case: '客户案例', product: '产品服务', soft: '资料下载'
}
export function typeNameOf(t) { return TYPE_NAMES[t] || t || '内容' }
export function fmtDate(t) { return t ? String(t).replace('T', ' ').slice(0, 16) : '' }
export function excerptOf(c, max = 110) {
const raw = c.seoDescription || String(c.content || '').replace(/<[^>]+>/g, ' ').replace(/\s+/g, ' ').trim()
return raw.length > max ? raw.slice(0, max) + '…' : (raw || '点击查看详情，获取完整内容。')
}
export function contentItems(contents, ref, max = 99) {
const list = Array.isArray(contents) ? contents : []
let arr = []
if (!ref || ref === 'all') {
arr = [...list]
} else if (ref === 'qa') {
arr = list.filter((c) => c.contentType === 'article')
if (!arr.length) arr = list.filter((c) => c.contentType === 'news')
if (!arr.length) arr = [...list]
} else {
arr = list.filter((c) => c.contentType === ref)
}
return arr.slice(0, max > 0 ? max : 99)
}
const siteBase = (ctx) => {
const id = ctx && ctx.site ? ctx.site.id : null
return id === null || id === undefined ? '/preview/site/0' : '/preview/site/' + id
}
export function absHref(ctx, href) {
if (!href) return '#'
if (/^(https?:)?\/\
return siteBase(ctx) + href
}
export function detailUrl(ctx, c) {
return siteBase(ctx) + '/detail/' + c.id
}
export const QA_TAGS = ['水电', '瓦工', '木工', '油漆', '设计', '预算', '验收', '环保', '经验']
export function askerOf(c) { return String(c.title).length % 2 === 0 ? '装修问答用户' : '热心工长' }
export function answerCount(c) { return ((Number(c.id) || 7) * 3) % 14 + 2 }
export function viewCount(c) { return ((Number(c.id) || 7) * 37) % 160 + 30 }
export function qaTagsOf(c, arr) {
const i = (arr || []).indexOf(c)
const k = i < 0 ? ((Number(c.id) || 7) % 97) : i
const a = QA_TAGS[(k * 3) % QA_TAGS.length]
const b = QA_TAGS[(k * 3 + 5) % QA_TAGS.length]
return a === b ? [a, QA_TAGS[(k * 3 + 1) % QA_TAGS.length]] : [a, b]
}
export function corpStatRows(ctx, max = 3) {
const by = (t) => contentItems(ctx.contents, t, 99).length
const set = []
if (by('product')) set.push({ label: '解决方案', value: by('product') + ' 项', icon: 'fa-solid fa-cube' })
```

```
import { computed } from 'vue'
import { resolveSkin } from './skin.js'
import BlockRenderer from './blocks/BlockRenderer.vue'
const props = defineProps({
site: { type: Object, required: true },
tpl: { type: Object, default: null },
content: { type: Object, required: true },
contents: { type: Array, default: () => [] },
pageConfig: { type: Object, default: () => ({}) }
})
const skinKey = computed(() => resolveSkin(props.tpl, props.site))
const isQa = computed(() => skinKey.value === 'qa')
const isPortal = computed(() => skinKey.value === 'portal')
const isCity = computed(() => skinKey.value === 'city')
const detailCfg = computed(() => (props.pageConfig && props.pageConfig.detail &&
Array.isArray(props.pageConfig.detail.blocks)) ? props.pageConfig.detail.blocks : null)
const detailBlocks = computed(() => (detailCfg.value || []).filter((b) => b && b.blockKey &&
b.enabled !== false))
const blockCtx = computed(() => ({ site: props.site, tpl: props.tpl, contents: props.contents, skinKey:
skinKey.value, pageKey: 'detail', type: props.content.contentType || 'all' }))
const QA_TAGS = ['水电', '瓦工', '木工', '油漆', '设计', '预算', '验收', '环保', '经验']
const questionDesc = computed(() => {
const raw = (props.content.seoDescription || '').trim()
return raw ? raw : ''
})
function askerOf(c) { return String(c.title).length % 2 === 0 ? '装修问答用户' : '热心工长' }
function answerCount(c) { return ((c.id || 7) * 3) % 14 + 2 }
function viewCount(c) { return ((c.id || 7) * 37) % 160 + 30 }
function qaTagsOf(c) {
const i = (c.id || 7) % 97
const a = QA_TAGS[i % QA_TAGS.length]
const b = QA_TAGS[(i * 3 + 5) % QA_TAGS.length]
return a === b ? [a, QA_TAGS[(i + 1) % QA_TAGS.length]] : [a, b]
}
const TYPE_NAMES = { all: '全部内容', news: '新闻动态', article: '深度文章', case: '客户案例', product: '
产品服务', soft: '资料下载' }
const QA_TYPE_NAMES = { all: '全部问答', news: '问答动态', article: '问答广场', case: '案例讨论', product:
'装修方案', soft: '装修知识库' }
const CITY_TYPE_NAMES = { all: '本地信息', news: '本地资讯', article: '城市话题', case: '民生专题',
product: '本地服务', soft: '便民资料' }
const typeLabel = computed(() => {
const base = TYPE_NAMES[props.content.contentType] || props.content.contentType || '内容'
if (isQa.value) return QA_TYPE_NAMES[props.content.contentType] || base
if (isCity.value) return CITY_TYPE_NAMES[props.content.contentType] || base
return base
})
const hasBody = computed(() => String(props.content.content || '').replace(/<[^>]+>/g,
'').trim().length > 0)
const sameType = computed(() => props.contents.filter((c) => c.contentType ===
props.content.contentType))
const idx = computed(() => sameType.value.findIndex((c) => c.id === props.content.id))
const prev = computed(() => (idx.value > 0 ? sameType.value[idx.value - 1] : null))
const next = computed(() => (idx.value >= 0 && idx.value < sameType.value.length - 1 ?
sameType.value[idx.value + 1] : null))
const related = computed(() =>
sameType.value.filter((c) => c.id !== props.content.id).slice(0, 3)
)
```

```
function excerpt(c) {
const raw = c.seoDescription || String(c.content || '').replace(/<[^>]+>/g, ' ').trim()
```

```
return set.slice(0, 3)
}
if (skinKey.value === 'corp-business') {
const set = []
if (productItems.value.length) set.push({ label: '解决方案', value: productItems.value.length + ' 项',
icon: 'fa-solid fa-briefcase' })
if (caseItems.value.length) set.push({ label: '经典案例', value: caseItems.value.length + ' 个', icon:
'fa-solid fa-award' })
if (newsItems.value.length) set.push({ label: '行业洞察', value: newsItems.value.length + ' 篇', icon:
'fa-solid fa-chart-line' })
return set
}
return []
})
const corpFeatures = [
{ icon: 'fa-solid fa-bolt', title: '极速部署', desc: '容器化架构，分钟级完成环境搭建，大幅缩短上线周期。' },
{ icon: 'fa-solid fa-shield-halved', title: '安全可靠', desc: '多层次安全防护体系，数据加密与合规审计，保障
业务稳定。' },
{ icon: 'fa-solid fa-chart-line', title: '智能分析', desc: 'AI 驱动的数据洞察，实时业务看板，辅助精准决策。
' }
]
const corpPoints = [
{ icon: 'fa-solid fa-circle-check', text: '依托会懂 GEO 全域营销系统，统一管理内容资产与品牌阵地' },
{ icon: 'fa-solid fa-arrows-rotate', text: '定期同步行业政策、产品与技术进展，保证信息鲜活可靠' },
{ icon: 'fa-solid fa-headset', text: '专业编辑团队持续供稿，7×24 在线支持站点运营' }
]
const serviceItems = computed(() => {
const fallback = [
{ icon: 'fa-solid fa-lightbulb', title: '战略咨询', desc: '市场洞察、商业模式设计、战略规划，为企业提供顶层
智慧支持。' },
{ icon: 'fa-solid fa-bullhorn', title: '品牌营销', desc: '品牌定位、整合营销传播、数字营销，提升品牌影响力和
市场占有率。' },
{ icon: 'fa-solid fa-hand-holding-dollar', title: '资本运作', desc: '投融资顾问、并购重组、上市辅导，助力企
业资本价值最大化。' }
]
const items = productItems.value.map((c) => ({ icon: 'fa-solid fa-gem', title: c.title, desc:
excerpt(c), link: c.id }))
return items.length >= 2 ? items.slice(0, 3) : fallback
})
const portalTags = [
{ icon: 'fa-solid fa-bolt', label: '光伏' },
{ icon: 'fa-solid fa-wind', label: '风电' },
{ icon: 'fa-solid fa-battery-full', label: '储能' },
{ icon: 'fa-solid fa-car', label: '新能源车' }
]
const sideFalls = ['《能源法》配套细则征求意见', '碳排放权交易管理办法修订', '分布式光伏发电补贴政策延续']
function sideItems(t, n) {
const arr = t === 'article' ? articleItems.value : t === 'news' ? newsItems.value : softItems.value
return arr.slice(0, n)
}
const stats4 = computed(() => {
const total = props.contents.length
return [
{ icon: 'fa-solid fa-file-lines', label: '收录资讯', value: total, unit: ' 篇' },
{ icon: 'fa-solid fa-newspaper', label: '行业动态', value: newsItems.value.length, unit: ' 篇' },
{ icon: 'fa-solid fa-book-open', label: '深度文章', value: articleItems.value.length, unit: ' 篇' },
{ icon: 'fa-solid fa-download', label: '资料下载', value: softItems.value.length, unit: ' 份' }
```

```
]
})
```

```
import { ref, computed, watch } from 'vue'
import { useRoute } from 'vue-router'
import { applySeo } from './preview/seo.js'
import { resolveSkin, SKIN_INFO, TYPE_META } from './preview/skin.js'
import PHome from './preview/PHome.vue'
import PList from './preview/PList.vue'
import PDtail from './preview/PDtail.vue'
import PPage from './preview/PPage.vue'
const route = useRoute()
const loading = ref(true)
const error = ref('')
const site = ref(null)
const tpl = ref(null)
const contents = ref([])
const content = ref(null)
const contentError = ref('')
const pageConfig = ref({})
const isTplDemo = computed(() => typeof route.name === 'string' && route.name.startsWith('tpl-'))
const siteId = computed(() => Number(route.params.id || 0))
const tplId = computed(() => Number(route.params.id || 0))
const listType = computed(() => route.params.type || 'all')
const pageKey = computed(() => route.params.page || 'about')
const tplLevel = computed(() => {
const n = route.name || ''
if (n === 'tpl-list') return 'list'
if (n === 'tpl-detail') return 'detail'
if (n === 'tpl-page') return 'page'
return 'home'
})
const demoSiteId = computed(() => (isTplDemo.value ? site.value && site.value.id : null))
const brandText = computed(() => {
if (isTplDemo.value) return (tpl.value && tpl.value.templateName) || '模板预览'
return (site.value && site.value.siteName) || '前台站点'
})
const footerDesc = computed(() => {
if (isTplDemo.value) return (tpl.value && `${tpl.value.templateType}类模板，支持完整前台页面层级与 SEO 标签输出。`) || ''
return (site.value && (site.value.tdkDescription || `欢迎访问 ${site.value.siteName}`)) || ''
})
const logoHref = computed(() =>
isTplDemo.value ? `/preview/template/${route.params.id}/home` : `/preview/site/${siteId.value}/home`
)
const TYPE_LABELS = { all: '全部内容', news: '新闻动态', article: '深度文章', case: '客户案例', product: '产品服务', soft: '资料下载' }
const CITIES = ['成都', '重庆', '西安', '武汉', '杭州', '广州', '北京', '上海', '深圳', '南京']
const currentCity = ref('成都')
const navOpen = ref(false)
const CITY_PROFILES = {
成都: { temp: '24°', weather: '多云转阵雨', humidity: '湿度 78%', air: '空气优 · 26', districts: ['天府新区', '锦江区', '高新区', '武侯区', '金牛区'], metro: '地铁 5 号线', place: '东门老街' },
重庆: { temp: '29°', weather: '阵雨转阴', humidity: '湿度 85%', air: '空气良 · 52', districts: ['两江新区', '渝中区', '江北区', '南岸区', '沙坪坝区'], metro: '轨道交通 18 号线', place: '洪崖洞' },
西安: { temp: '22°', weather: '晴间多云', humidity: '湿度 46%', air: '空气优 · 38', districts: ['高新区', '雁塔区', '碑林区', '未央区', '长安区'], metro: '地铁 8 号线', place: '大唐不夜城' },
武汉: { temp: '27°', weather: '多云有阵雨', humidity: '湿度 76%', air: '空气良 · 61', districts: ['东湖高新区', '武昌区', '江汉区', '汉阳区', '洪山区'], metro: '地铁 11 号线', place: '江汉路步行街' },
```

```
import { ref, reactive, onMounted } from 'vue'
import { ElMessage, ElMessageBox } from 'element-plus'
import { Plus } from '@element-plus/icons-vue'
import { userPage, userSave, userUpdate, userDelete, userResetPassword } from '../../api'
const roleDict = { ADMIN: '管理员', USER: '运营人员' }
const list = ref([])
const total = ref(0)
const loading = ref(false)
const saving = ref(false)
const visible = ref(false)
const query = reactive({ keyword: '', role: '', status: null, pageNum: 1, pageSize: 10 })
const emptyForm = () => ({ id: null, username: '', password: '123456', realName: '', email: '', phone:
'', role: 'USER', status: 1 })
const form = ref(emptyForm())
function roleText(role) {
return roleDict[role] || role || '-'
}
async function load() {
loading.value = true
try {
const params = { ...query, status: query.status ?? undefined }
const data = await userPage(params)
list.value = data.records || []
total.value = data.total || 0
} finally {
loading.value = false
}
}
function resetQuery() {
query.keyword = ''
query.role = ''
query.status = null
query.pageNum = 1
load()
}
function openDialog(row) {
form.value = row ? { ...row, password: '' } : emptyForm()
visible.value = true
}
async function save() {
if (!form.value.username) {
ElMessage.warning('请填写用户名')
return
}
saving.value = true
try {
if (form.value.id) {
const { password, ...rest } = form.value
await userUpdate(rest)
} else {
await userSave(form.value)
```

```
export const REGISTRY = {
hero: {
label: '主视觉 Banner',
desc: '官网首页顶部主视觉，展示标题/副标题/按钮与数据条；统计项缺失时自动按站点内容数量生成。',
icon: 'fa-solid fa-wand-magic-sparkles',
defaults: {
tag: '数字营销 · 全域内容运营',
title: '{{site.tdkTitle}}',
sub: '{{site.tdkDescription}}',
bgImage: '',
primaryText: '了解更多',
primaryHref: '/list/product',
secondaryText: '',
secondaryHref: '',
items: []
},
fields: [
{ key: 'tag', label: '角标文案', type: 'text', placeholder: '如 数字营销 · 全域内容运营' },
{ key: 'title', label: '主标题', type: 'text', placeholder: '支持 {{site.tdkTitle}} 占位' },
{ key: 'sub', label: '副标题', type: 'textarea', placeholder: '支持 {{site.tdkDescription}} 占位' },
{ key: 'bgImage', label: '背景图（可选）', type: 'image', placeholder: '留空使用模板默认渐变背景' },
{ key: 'primaryText', label: '主按钮文案', type: 'text' },
{ key: 'primaryHref', label: '主按钮链接', type: 'text', placeholder: '/list/product 或 /preview/...' },
{ key: 'secondaryText', label: '次按钮文案', type: 'text', placeholder: '留空隐藏' },
{ key: 'secondaryHref', label: '次按钮链接', type: 'text' },
{ key: 'items', label: '数据统计行（留空自动统计）', type: 'list', itemFields: [
{ key: 'icon', label: '图标', type: 'text', placeholder: 'fa-solid fa-*' },
{ key: 'label', label: '标签', type: 'text' },
{ key: 'value', label: '数值文案', type: 'text' }
] }
]
},
textHtml: {
label: '富文本内容',
desc: '自由排版段落/图文区域，HTML 内容支持 {{site.siteName}} 等站点字段占位。',
icon: 'fa-solid fa-align-left',
defaults: {
kicker: '',
title: '',
html: '',
align: 'left'
},
fields: [
{ key: 'kicker', label: '小标题（眉题）', type: 'text' },
{ key: 'title', label: '区块标题', type: 'text' },
{ key: 'html', label: 'HTML 正文', type: 'textarea' },
{ key: 'align', label: '对齐方式', type: 'text', placeholder: 'left / center' }
]
},
features: {
```

```
import { h } from 'vue'
import { ElInput, ElInputNumber, ElSwitch, ElSelect, ElOption, ElButton } from 'element-plus'
const defaultForType = (t) => (t === 'number' ? 0 : t === 'switch' ? false : t === 'list' ? [] : '')
const FieldControl = {
name: 'FieldControl',
props: {
field: { type: Object, required: true },
modelValue: { default: null },
options: { type: Array, default: () => [] }
},
emits: ['update:modelValue'],
methods: {
set(v) {
this.$emit('update:modelValue', v)
}
},
render() {
const f = this.field
if (!f || !f.type) return null
const set = (v) => this.set(v)
const placeholder = f.placeholder || ''
switch (f.type) {
case 'textarea': {
return h(ElInput, {
modelValue: this.modelValue == null ? '' : String(this.modelValue),
type: 'textarea', rows: f.rows || 3, placeholder,
'onUpdate:modelValue': (v) => set(v ?? '')
})
}
case 'number': {
return h(ElInputNumber, {
modelValue: this.modelValue == null ? 0 : this.modelValue,
style: 'width: 100%', controlsPosition: 'right', placeholder,
'onUpdate:modelValue': (v) => set(v == null ? 0 : v)
})
}
case 'switch': {
return h(ElSwitch, {
modelValue: !!this.modelValue,
'onUpdate:modelValue': (v) => set(!!v)
})
}
case 'image': {
const img = this.modelValue
? h('img', { class: 'dv-img-preview', src: String(this.modelValue), alt: 'preview' })
: null
return h('div', { class: 'dv-img-field' }, [
h(ElInput, {
modelValue: this.modelValue == null ? '' : String(this.modelValue),
placeholder: placeholder || '图片 URL', clearable: true,
```

```
}
public void deleteKeyword(Long id) {
keywordMapper.deleteById(id);
}
public PageResult<Map<String, Object>> pageAlert(Long keywordId, String level, Integer status, int
pageNum, int pageSize) {
LambdaQueryWrapper<MonitorAlert> q = Wrappers.<MonitorAlert>lambdaQuery()
.eq(keywordId != null, MonitorAlert::getKeywordId, keywordId)
.eq(StringUtils.hasText(level), MonitorAlert::getAlertLevel, level)
.eq(status != null, MonitorAlert::getAlertStatus, status)
.orderByDesc(MonitorAlert::getId);
Page<MonitorAlert> p = alertMapper.selectPage(new Page<>(pageNum, pageSize), q);
List<Map<String, Object>> list = new ArrayList<>();
for (MonitorAlert a : p.getRecords()) {
Map<String, Object> m = new HashMap<>();
m.put("id", a.getId());
m.put("keywordId", a.getKeywordId());
m.put("alertTitle", a.getAlertTitle());
m.put("alertContent", a.getAlertContent());
m.put("alertSource", a.getAlertSource());
m.put("alertUrl", a.getAlertUrl());
m.put("alertLevel", a.getAlertLevel());
m.put("alertStatus", a.getAlertStatus());
m.put("handleResult", a.getHandleResult());
m.put("workOrderId", a.getWorkOrderId());
m.put("createTime", a.getCreateTime());
MonitorKeyword kw = a.getKeywordId() == null ? null : keywordMapper.selectById(a.getKeywordId());
m.put("keyword", kw == null ? "-" : kw.getKeyword());
list.add(m);
}
return new PageResult<>(p.getTotal(), list);
}
public void handleAlert(Long id, Integer status, String result) {
if (status == null || (status != 1 && status != 2)) {
throw new BusinessException("非法处理状态，仅支持 1=处理中 / 2=已处理");
}
MonitorAlert exist = alertMapper.selectById(id);
if (exist == null) {
throw new BusinessException("预警不存在");
}
if (exist.getAlertStatus() != null && exist.getAlertStatus() == 3) {
throw new BusinessException("该预警已转工单，请前往工单中心跟进，勿直接修改处理状态");
}
if (exist.getAlertStatus() != null && exist.getAlertStatus() == 2) {
throw new BusinessException("该预警已处理完成，无需重复操作");
}
MonitorAlert alert = new MonitorAlert();
alert.setId(id);
alert.setAlertStatus(status);
if (StringUtils.hasText(result)) {
alert.setHandleResult(result);
```

```
export const customerChangeStatus = (id, status) => request.put('/api/crm/customer/' + id + '/status',
null, { params: { status } })
export const servicePage = (params) => request.get('/api/crm/service/page', { params })
export const serviceSave = (data) => request.post('/api/crm/service', data)
export const serviceDelete = (id) => request.delete('/api/crm/service/' + id)
export const orderPage = (params) => request.get('/api/crm/order/page', { params })
export const orderSave = (data) => request.post('/api/crm/order', data)
export const orderUpdate = (data) => request.put('/api/crm/order', data)
export const orderDetail = (id) => request.get('/api/crm/order/' + id)
export const orderFinish = (id, result) => request.put('/api/crm/order/' + id + '/finish', null,
{ params: { result } })
export const orderAccept = (id, remark) => request.put('/api/crm/order/' + id + '/accept', null,
{ params: { remark } })
export const orderAssign = (id, assignee, remark) => request.put('/api/crm/order/' + id + '/assign',
null, { params: { assignee, remark } })
export const orderClose = (id, remark) => request.put('/api/crm/order/' + id + '/close', null, { params:
{ remark } })
export const orderDelete = (id) => request.delete('/api/crm/order/' + id)
export const orderUsers = () => request.get('/api/crm/order/users')
export const reportPage = (params) => request.get('/api/ai/report/page', { params: { pageNum:
params.pageNum, pageSize: params.pageSize, type: params.type || params.reportType || '' } })
export const reportGet = (id) => request.get('/api/ai/report/' + id)
export const reportGenerate = (params) => request.post('/api/ai/report/generate', null, { params })
export const reportDelete = (id) => request.delete('/api/ai/report/' + id)
export const workflowPage = (params) => request.get('/api/ai/workflow/page', { params })
export const workflowSave = (data) => request.post('/api/ai/workflow', data)
export const workflowUpdate = (data) => request.put('/api/ai/workflow', data)
export const workflowRun = (id) => request.post('/api/ai/workflow/' + id + '/run')
export const workflowDelete = (id) => request.delete('/api/ai/workflow/' + id)
export const knowledgePage = (params) => request.get('/api/ai/knowledge/page', { params })
export const knowledgeGet = (id) => request.get('/api/ai/knowledge/' + id)
export const knowledgeSave = (data) => request.post('/api/ai/knowledge', data)
export const knowledgeUpdate = (data) => request.put('/api/ai/knowledge', data)
export const knowledgeDelete = (id) => request.delete('/api/ai/knowledge/' + id)
export const assistantAsk = (data) => request.post('/api/ai/assistant', data)
export const assistantChat = (data) => request.post('/api/ai/assistant/chat', data)
export const assistantHistory = (sessionId) => request.get('/api/ai/assistant/history', { params:
{ sessionId } })
export const assistantClear = (sessionId) => request.delete('/api/ai/assistant/clear', { params:
{ sessionId } })
export const assistantSessions = () => request.get('/api/ai/assistant/sessions')
export const aiOverview = () => request.get('/api/ai/overview')
export const aiModelStatus = () => request.get('/api/ai/model-status')
export const userPage = (params) => request.get('/api/system/user/page', { params })
export const userSave = (data) => request.post('/api/system/user', data)
export const userUpdate = (data) => request.put('/api/system/user', data)
export const userDelete = (id) => request.delete('/api/system/user/' + id)
export const userResetPassword = (id, password) => request.put('/api/system/user/' + id + '/reset-
password', null, { params: { password } })
export const logPage = (params) => request.get('/api/system/log/page', { params })
export const logDetail = (id) => request.get('/api/system/log/' + id)
export const logClean = (data) => request.post('/api/system/log/clean', data)
export const configList = () => request.get('/api/system/config/list')
export const configPage = (params) => request.get('/api/system/config/page', { params })
export const configSave = (data) => request.post('/api/system/config', data)
export const configUpdate = (data) => request.put('/api/system/config', data)
export const configDelete = (id) => request.delete('/api/system/config/' + id)
```

```
export const uploadFile = (file) => {
const form = new FormData()
```

```
localStorage.setItem(SESSION_KEY, sessionId.value)
welcomeMessages()
await loadSessions()
await scrollBottom()
}
async function switchSession(sid) {
if (sid === sessionId.value) return
sessionId.value = sid
localStorage.setItem(SESSION_KEY, sid)
messages.value = []
await loadHistory(sid)
await loadSessions()
}
async function clearCurrent() {
await ElMessageBox.confirm('确认清空当前会话的消息记录？', '清空会话', { type: 'warning' })
await assistantClear(sessionId.value)
welcomeMessages()
await loadSessions()
await scrollBottom()
}
async function send() {
const text = input.value.trim()
if (!text || sending.value) return
ensureSession()
messages.value.push({ role: 'user', content: text })
input.value = ''
sending.value = true
await scrollBottom()
try {
const res = await assistantChat({ message: text, sessionId: sessionId.value })
messages.value.push({ role: 'assistant', content: res.answer || res.content || '已收到你的问题。' })
await loadSessions()
} catch {
messages.value.push({ role: 'assistant', content: '请求失败，请检查后端服务后重试。' })
} finally {
sending.value = false
await scrollBottom()
}
}
function shortTime(t) {
if (!t) return ''
const s = String(t).replace('T', ' ').slice(5, 16)
return s
}
onMounted(async () => {
ensureSession()
await loadSessions()
await loadHistory(sessionId.value)
await loadModelStatus()
})
```

```
}
}
private int seedConfigRows() {
Map<String, String[]> rows = new LinkedHashMap<>();
rows.put("sys.upload.dir", new String[]{"./uploads", "上传目录", "text", "文件上传保存目录"});
rows.put("sys.upload.maxsize", new String[]{"20", "上传大小限制(MB)", "number", "单文件上传大小上限，单位
MB"});
rows.put("sys.ai.enabled", new String[]{"true", "AI 服务开关", "boolean", "是否启用 AI 能力中心功能"});
rows.put("sys.site.defaultDomain", new String[]{"http://localhost:8080", "分发默认域名", "text", "全域分
发对外链接默认域名前缀"});
rows.put("sys.log.keepDays", new String[]{"90", "日志保留天数", "number", "操作日志清理默认保留最近 N 天，0
表示不自动清理"});
rows.put("sys.ai.apiKey", new String[]{"", "AI 接口密钥", "password", "调用外部大模型接口的密钥（敏感配置，
掩码展示）"});
int added = 0;
try (Connection conn = dataSource.getConnection()) {
for (Map.Entry<String, String[]> e : rows.entrySet()) {
String key = e.getKey();
String[] v = e.getValue();
long cnt;
try (PreparedStatement ps = conn.prepareStatement("SELECT COUNT(*) AS C FROM sys_config WHERE
config_key = ?")) {
ps.setString(1, key);
try (ResultSet rs = ps.executeQuery()) {
rs.next();
cnt = rs.getLong("C");
}
}
Timestamp now = Timestamp.valueOf(LocalDateTime.now());
if (cnt > 0) {
try (PreparedStatement ps = conn.prepareStatement("UPDATE sys_config SET config_type = ?, config_name
= ?, remark = ?, update_time = ? WHERE config_key = ?")) {
ps.setString(1, v[2]);
ps.setString(2, v[1]);
ps.setString(3, v[3]);
ps.setTimestamp(4, now);
ps.setString(5, key);
ps.executeUpdate();
}
} else {
try (PreparedStatement ps = conn.prepareStatement("INSERT INTO sys_config (config_key, config_value,
config_name, config_type, remark, create_time, update_time) VALUES (?, ?, ?, ?, ?, ?, ?)")) {
ps.setString(1, key);
ps.setString(2, v[0]);
ps.setString(3, v[1]);
ps.setString(4, v[2]);
ps.setString(5, v[3]);
ps.setTimestamp(6, now);
ps.setTimestamp(7, now);
ps.executeUpdate();
}
added++;
}
}
} catch (Exception e) {
log.warn("配置种子执行失败: {}", e.getMessage());
}
```

```
}
Map<String, Long> contentMap = countByDay("content_info");
Map<String, Long> distMap = countByDay("distribute_record");
Map<String, Long> exposeMap = exposeByDay();
List<Long> content = new ArrayList<>();
List<Long> distribute = new ArrayList<>();
List<Long> expose = new ArrayList<>();
for (String d : dates) {
content.add(contentMap.getOrDefault(d, 0L));
distribute.add(distMap.getOrDefault(d, 0L));
expose.add(exposeMap.getOrDefault(d, 0L));
}
Map<String, Object> m = new HashMap<>();
m.put("dates", dates);
m.put("content", content);
m.put("distribute", distribute);
m.put("expose", expose);
return Result.success(m);
}
@GetMapping("/distribution")
public Result<List<Map<String, Object>>> distribution() {
List<Map<String, Object>> rows = jdbcTemplate.queryForList(
"SELECT p.platform_name AS pname, COUNT(r.id) AS cnt, COALESCE(SUM(r.expose_count),0) AS exp, " +
"COALESCE(SUM(r.read_count),0) AS rd " +
"FROM distribute_record r LEFT JOIN distribute_platform p ON r.platform_id = p.id " +
"GROUP BY p.platform_name");
List<Map<String, Object>> list = new ArrayList<>();
for (Map<String, Object> row : rows) {
Map<String, Object> item = new HashMap<>();
item.put("platformName", String.valueOf(row.get("pname") == null ? "未命名平台" : row.get("pname")));
item.put("count", toLong(row.get("cnt")));
item.put("expose", toLong(row.get("exp")));
item.put("read", toLong(row.get("rd")));
list.add(item);
}
return Result.success(list);
}
private Map<String, Long> countByDay(String table) {
Map<String, Long> result = new HashMap<>();
String sql = "SELECT FORMATDATETIME(create_time,'MM-dd') AS d, COUNT(*) AS c FROM " + table
+ " WHERE create_time >= DATEADD('DAY',-6,CURRENT_DATE) GROUP BY FORMATDATETIME(create_time,'MM-dd')";
for (Map<String, Object> row : jdbcTemplate.queryForList(sql)) {
result.put(String.valueOf(row.get("d")), toLong(row.get("c")));
}
return result;
}
private Map<String, Long> exposeByDay() {
Map<String, Long> result = new HashMap<>();
String sql = "SELECT FORMATDATETIME(create_time,'MM-dd') AS d, COALESCE(SUM(expose_count),0) AS e FROM
distribute_record "
+ " WHERE create_time >= DATEADD('DAY',-6,CURRENT_DATE) GROUP BY FORMATDATETIME(create_time,'MM-dd')";
```

```
throw new com.geo.common.BusinessException("站点名称与域名必填");
}
if (siteInfoMapper.selectCount(Wrappers.<SiteInfo>lambdaQuery().eq(SiteInfo::getDomain,
site.getDomain())) > 0) {
throw new com.geo.common.BusinessException("该域名已存在");
}
site.setId(null);
site.setCreateBy(currentUser());
site.setCreateTime(LocalDateTime.now());
site.setUpdateTime(LocalDateTime.now());
if (site.getStatus() == null) {
site.setStatus(1);
}
siteInfoMapper.insert(site);
}
public void updateSite(SiteInfo site) {
site.setUpdateBy(currentUser());
site.setUpdateTime(LocalDateTime.now());
siteInfoMapper.updateById(site);
}
public void deleteSite(Long id) {
siteInfoMapper.deleteById(id);
}
public void changeSiteStatus(Long id, int status) {
if (status != 0 && status != 1) {
throw new com.geo.common.BusinessException("非法状态值，仅支持 0=停用 / 1=启用");
}
SiteInfo exist = siteInfoMapper.selectById(id);
if (exist == null) {
throw new com.geo.common.BusinessException("站点不存在");
}
if (exist.getStatus() != null && exist.getStatus() == status) {
return;
}
SiteInfo patch = new SiteInfo();
patch.setId(id);
patch.setStatus(status);
patch.setUpdateBy(currentUser());
patch.setUpdateTime(LocalDateTime.now());
siteInfoMapper.updateById(patch);
}
public void updateSeo(Long id, String tdkTitle, String tdkKeywords, String tdkDescription) {
SiteInfo site = new SiteInfo();
site.setId(id);
site.setTdkTitle(tdkTitle);
site.setTdkKeywords(tdkKeywords);
site.setTdkDescription(tdkDescription);
site.setUpdateTime(LocalDateTime.now());
siteInfoMapper.updateById(site);
}
public PageResult<SiteTemplate> pageTemplate(String keyword, int pageNum, int pageSize) {
```

```
seedTemplateTdkDefaults();
fixLegacySite4();
Long chengduSiteId = ensureChengduCitySite();
if (chengduSiteId != null) {
seedChengduContents(chengduSiteId);
}
cleanDetailUnknownBlocks();
migrateStatusDicts();
ensureCrmBatch2Tables();
ensureAiBatch3Tables();
Long bizSiteId = ensureCorpBusinessTemplate();
if (bizSiteId != null) {
seedBizContents(bizSiteId);
}
log.info("BootstrapRunner 引导完成");
}
private void ensurePageSchemaTable() {
String sql = "CREATE TABLE IF NOT EXISTS page_schema (" +
"  id BIGINT AUTO_INCREMENT PRIMARY KEY," +
"  template_id BIGINT," +
"  site_id BIGINT," +
"  page_key VARCHAR(32) NOT NULL," +
"  blocks CLOB," +
"  status TINYINT DEFAULT 1," +
"  create_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP," +
"  update_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP," +
"  CHECK (template_id IS NOT NULL OR site_id IS NOT NULL)" +
")";
try (Connection conn = dataSource.getConnection(); Statement st = conn.createStatement()) {
st.execute(sql);
log.info("page_schema 表已就绪(IF NOT EXISTS)");
} catch (Exception e) {
log.warn("page_schema 建表失败: {}", e.getMessage());
}
}
private void fixLegacySite4() {
List<SiteInfo> legacy = siteInfoMapper.selectList(Wrappers.<SiteInfo>lambdaQuery()
.eq(SiteInfo::getSiteName, SITE4_NAME)
.eq(SiteInfo::getTemplateId, 4L));
if (legacy.isEmpty()) {
SiteInfo byId = siteInfoMapper.selectById(4L);
if (byId != null && byId.getTemplateId() != null && byId.getTemplateId() == 4L) {
legacy = List.of(byId);
}
}
boolean changed = false;
for (SiteInfo s : legacy) {
String type = s.getSiteType();
Long tplId = s.getTemplateId();
if ("门户".equals(type) && tplId != null && tplId == 2L) {
```