**日志链路**

**前言**

每个请求会生成一个trackID，根据trackId可以查询到此次请求的所有日志

**实现**

**添加pox依赖**

pom.xml

|  |
| --- |
| XML <dependency>  <groupId>com.sleepycat</groupId>  <artifactId>je</artifactId>  <version>${je.version}</version> </dependency> |

**修改logbak日志配置**

logback-spring.xml

|  |
| --- |
| XML <appender name="BERKELEY\_DB" class="org.disco.core.linklog.LogBackBerkeleyDbAppender">  <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">  <pattern>%m</pattern>  </encoder>  </appender> |

**添加过滤器，给每个请求生成一个TrackID**

LinkLogFilter

|  |
| --- |
| Java @Slf4j public class LinkLogFilter implements Filter {   @Override  public void init(FilterConfig filterConfig) throws ServletException {  }   @Override  public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse, FilterChain filterChain) throws IOException, ServletException {  long start = System.currentTimeMillis();  String trackId = null;  if (servletRequest instanceof HttpServletRequest) {  //先从请求头中获取trackId  HttpServletRequest request = (HttpServletRequest) servletRequest;  trackId = request.getHeader(Globals.TRACKID);  if (StringUtils.isBlank(trackId)) {  trackId = UniqueUtil.getUUID8();  }  MDC.put(Globals.TRACKID, trackId);  //打印参数日志  log.info("请求开始,路径:{},方式:{},参数类型:{},参数:{}", RequestUtil.getSiteRoot() + request.getRequestURI(), request.getMethod(), request.getHeader("Content-Type"), RequestUtil.getRequestParams(request));  }  if (servletResponse instanceof HttpServletResponse) {  HttpServletResponse response = (HttpServletResponse) servletResponse;  response.setHeader(Globals.TRACKID, trackId);  }  filterChain.doFilter(servletRequest, servletResponse);  log.info("请求结束,耗时:{}ms..", System.currentTimeMillis() - start);  MDC.clear();  }   @Override  public void destroy() {   } } |

**将日志写入BerkeleyDb**

LogBackBerkeleyDbAppender.java

|  |
| --- |
| Java import ch.qos.logback.classic.spi.ILoggingEvent; import ch.qos.logback.core.UnsynchronizedAppenderBase; import ch.qos.logback.core.encoder.Encoder; import lombok.Getter; import lombok.Setter; import lombok.SneakyThrows; import lombok.extern.slf4j.Slf4j; import org.apache.commons.lang3.StringUtils; import org.disco.core.Globals; import org.disco.core.Values; import org.slf4j.MDC;  @Slf4j @Getter @Setter public class LogBackBerkeleyDbAppender extends UnsynchronizedAppenderBase<ILoggingEvent> {   protected Encoder<ILoggingEvent> encoder;   @Override  public void start() {  super.start();  //关闭服务的时候关闭日志数连接  Runtime.getRuntime().addShutdownHook(new Thread(LogBackBerkeleyDbHelper::closeEnv));  }   @SneakyThrows  @Override  protected void append(ILoggingEvent event) {  String trackId = MDC.get(Globals.TRACKID);  if (!Values.getInstance().getLinklogBerkeleydbEnable() || StringUtils.isBlank(trackId) || event == null) {  return;  }  LogBackBerkeleyDbHelper.getInstance().putTodayLog(trackId, new LogBackLinkLogInfo(event,encoder));  } } |

LogBackBerkeleyDbHelper.java

|  |
| --- |
| Java import com.sleepycat.je.\*; import lombok.SneakyThrows; import lombok.extern.slf4j.Slf4j; import org.apache.commons.collections4.CollectionUtils; import org.apache.commons.lang3.StringUtils; import org.apache.commons.lang3.time.DateFormatUtils; import org.apache.commons.lang3.time.DateUtils; import org.disco.core.Globals; import org.disco.core.Values; import org.disco.core.utils.DateUtil; import org.disco.core.utils.SerializationUtil;  import java.io.File; import java.io.UnsupportedEncodingException; import java.time.LocalDateTime; import java.util.ArrayList; import java.util.Comparator; import java.util.Date; import java.util.List; import java.util.stream.Collectors;  @Slf4j public class LogBackBerkeleyDbHelper {   private static volatile LogBackBerkeleyDbHelper instance;   private final static String LOG\_DIR = Globals.APP\_FILE\_DIR + "log/";  private final static String PATTERN = "yyyyMMdd";  private final static String CHARSETNAME = "UTF-8";   /\*\*  \* 数据源  \*/  private static Environment env = null;   /\*\*  \* 数据库配置  \*/  private static DatabaseConfig dbConfig = null;   /\*\*  \* 数据库  \*/  private static Database database = null;   private LogBackBerkeleyDbHelper() {  initEnvironment();  }   public static LogBackBerkeleyDbHelper getInstance() {  LogBackBerkeleyDbHelper result = instance;  if (result == null) {  synchronized (LogBackBerkeleyDbHelper.class) {  result = instance;  if (result == null) {  instance = result = new LogBackBerkeleyDbHelper();  }  }  }  return result;  }   /\*\*  \* 关闭链接  \*/  public static void closeEnv() {  if (database != null) {  database.close();  }  if (env != null) {  log.info("关闭LogbackBerkeleyDB....");  env.cleanLog();  env.close();  }  }   /\*\*  \* 初始化连接  \*/  private void initEnvironment() {  EnvironmentConfig configuration = new EnvironmentConfig().setAllowCreate(true).setTransactional(true);  File envHome = new File(LOG\_DIR+Values.getInstance().getAppName());  if (!envHome.exists()) {  envHome.mkdirs();  }  env = new Environment(envHome, configuration);  dbConfig = new DatabaseConfig()  .setAllowCreate(true)  .setSortedDuplicates(true);  log.info("初始化LogbackBerkeleyDB....");  }   /\*\*  \* 向数据库中写入记录  \*/  public void putTodayLog(String key, LogBackLinkLogInfo info) throws UnsupportedEncodingException {  String dbName = DateFormatUtils.format(new Date(), PATTERN);  if (database == null || !StringUtils.equals(dbName, database.getDatabaseName())) {  //防止被误关闭  if (env == null) {  initEnvironment();  }  //创建前关闭之前的数据库  if (database != null) {  database.close();  //日志库变更可触发清除过期日志任务  clearHistory();  }  //使用Environment与DatabaseConfig打开Database  database = env.openDatabase(null, dbName, dbConfig);  }  // 设置key/value,注意DatabaseEntry内使用的是bytes数组  DatabaseEntry theKey = new DatabaseEntry(key.getBytes(CHARSETNAME));  DatabaseEntry theData = new DatabaseEntry(SerializationUtil.serialize(info));  database.put(null, theKey, theData);  env.sync();  }   /\*\*  \* 获取指定key的数据  \*  \* @param key  \* @return  \*/  @SneakyThrows  public List<LogBackLinkLogInfo> getKey(String key, LocalDateTime start, LocalDateTime end){  List<LogBackLinkLogInfo> list = new ArrayList<>();  List<String> databaseNames = env.getDatabaseNames();  if (CollectionUtils.isEmpty(databaseNames)) {  return list;  }  //排除开始时间以外的  if (start != null) {   String startTime = DateUtil.formatAsLocalDateTime(start, PATTERN);  databaseNames = databaseNames.stream().filter(databaseName ->  Long.parseLong(databaseName) >= Long.parseLong(startTime)  ).collect(Collectors.toList());  }  //排除结束时间以外的  if (end != null) {  String endTime = DateUtil.formatAsLocalDateTime(end, PATTERN);  databaseNames = databaseNames.stream().filter(databaseName ->  Long.parseLong(databaseName) <= Long.parseLong(endTime)  ).collect(Collectors.toList());  }   //遍历所有库  for (String databaseName : databaseNames) {  Database db = null;  Cursor cursor = null;  try {  db = env.openDatabase(null, databaseName, dbConfig);  if (database.count() < 1) {  continue;  }  CursorConfig cursorConfig = new CursorConfig();  cursorConfig.setReadUncommitted(true);  cursor = db.openCursor(null, cursorConfig);  if(key==null) {  key="";  }  DatabaseEntry databaseKey = new DatabaseEntry(key.getBytes(CHARSETNAME));  DatabaseEntry databaseValue = new DatabaseEntry();  OperationStatus res = cursor.getSearchKey(databaseKey, databaseValue, LockMode.DEFAULT);  while (res == OperationStatus.SUCCESS) {  list.add(SerializationUtil.convert(databaseValue.getData(),LogBackLinkLogInfo.class));  res = cursor.getNextDup(databaseKey, databaseValue, LockMode.DEFAULT);  }  } finally {  if (cursor != null) {  cursor.close();  }  if (db != null) {  db.close();  }  }  }  list.sort(Comparator.comparing(LogBackLinkLogInfo::getTimeStamp));  return list;  }   /\*\*  \* 清除过期数据  \*/  public void clearHistory() {  //获取保留日期内的最小天数  Date date = DateUtils.addDays(new Date(), -Values.getInstance().getLinklogBerkeleydbLogHistoryDay());  //获取基准库数字  Long indexDbName = Long.parseLong(DateFormatUtils.format(date, PATTERN));  log.info("清理LogbackBerkeleyDB过期数据,基准日期{}....", indexDbName);  List<String> databaseNames = env.getDatabaseNames();  for (String databaseName : databaseNames) {  //根据当前日志规则 当前库<基准库 则认为日志已失效 删除数据库  if (Long.parseLong(databaseName) < indexDbName) {  log.info("清理LogbackBerkeleyDB过期数据,清除库:{}....", databaseName);  env.removeDatabase(null, databaseName);  }  }  }  } |

LogBackLinkLogInfo.java

|  |
| --- |
| Java @Getter @Setter public class LogBackLinkLogInfo implements Serializable {   @JsonIgnore  private Long timeStamp;  private String level;  private String threadName;  private String loggerName;  private String message;  private String trackId;   public LogBackLinkLogInfo(ILoggingEvent event, Encoder<ILoggingEvent> encoder) {  this.timeStamp = event.getTimeStamp();  this.level = event.getLevel().levelStr;  this.threadName = event.getThreadName();  this.loggerName = event.getLoggerName();  Map<String, String> mdcPropertyMap = event.getMDCPropertyMap();  this.trackId = Objects.isNull(mdcPropertyMap) ? null : mdcPropertyMap.get(Globals.TRACKID);  this.message = new String(encoder.encode(event));  }   public String getLogTime() {  return String.valueOf(timeStamp);  } } |

**提供一个日志链路查询接口**

LogLinkController.java

|  |
| --- |
| Java @Slf4j @RestController @Tag(name = "日志链路") @RequestMapping("/common/log/link") public class LogLinkController {   @Autowired  private Values values;   @Resource  private LogLinkClient logLinkClient;   @PostMapping(value = "/track")  @Operation(summary = "追踪日志")  //@PreAuthorize("@ss.hasPerm('d:common:link:track')")  public Result doTrack(@RequestBody LogLinkVo logLinkVo, HttpServletResponse response) {  if(logLinkVo.getAppName()==null || logLinkVo.getAppName().equals(values.getAppName())){  return Result.buildSuccess(LogBackBerkeleyDbHelper.getInstance().getKey(logLinkVo.getTrackId(),  logLinkVo.getStartTime(), logLinkVo.getEndTime()));  } else if (values.getAppCluster().containsKey(logLinkVo.getAppName())){  String baseUrl = values.getAppCluster().get(logLinkVo.getAppName());  ForestResponse<Result> fr = logLinkClient.logLinkTrack(baseUrl, logLinkVo);  ResponseUtil.writerResult(response, fr.getResult(), fr.getStatusCode(), true);  }  return Result.buildSuccess();  }  } |

LogLinkVo.java

|  |
| --- |
| Java @Data @AllArgsConstructor @NoArgsConstructor @Builder public class LogLinkVo {  private String appName;  private String trackId;  private LocalDateTime startTime;  private LocalDateTime endTime; } |

**前端展示日志的vue代码**

logLink/index.vue

|  |
| --- |
| JavaScript <template>  <div class="log-container">  <el-card>  <!-- <el-form ref="form" :model="form" label-width="80px">-->  <el-form :model="form" :inline="true" label-width="80px">  <el-form-item label="应用" :span="6">  <el-select v-model="form.appName" placeholder="">  <el-option label="SaaS" value="shipber" />  <el-option label="Op" value="shipberOp" />  </el-select>  </el-form-item>  <el-form-item label="TrackId" :span="10">  <el-input v-model="form.trackId"></el-input>  </el-form-item>  <el-form-item label="时间">  <el-col :span="11">  <el-date-picker type="date" placeholder="startTime" value-format="YYYY-MM-DD HH:mm:ss"  v-model="form.startTime" style="width: 100%;"></el-date-picker>  </el-col>  <span class="spacing">-</span>  <el-col :span="11">  <el-date-picker type="date" placeholder="endTime" value-format="YYYY-MM-DD HH:mm:ss" v-model="form.endTime"  style="width: 100%;"></el-date-picker>  </el-col>  </el-form-item>  <el-form-item>  <el-button type="primary" @click="showLog">search</el-button>  </el-form-item>  <el-form-item style="float: right">  <el-switch v-model="wrap" active-color="#178AFC" inactive-color="#EAECF0">  </el-switch>  </el-form-item>  </el-form>  </el-card>   <el-card class="box-card infinite-list-wrapper">  <ul class="list" :style="{height: logHeight}" style="overflow:auto">  <li v-for="(log) in logs.list" :key="log.id" class="text item" :class="{ 'wrap': wrap }">  {{ formartTime(log.logTime) }}  &nbsp;[{{ log.trackId }}]&nbsp;  <span v-if="log.level == 'INFO'" style="color: limegreen;font-weight: bold;">{{ log.level }}</span>  <span v-else-if="log.level == 'WARN'" style="color: orange;font-weight: bold;">{{ log.level }}</span>  <span v-else-if="log.level == 'ERROR'" style="color: red;font-weight: bold;">{{ log.level }}</span>  <span v-else style="font-weight: bold;">{{ log.level }}</span>  &nbsp;[{{ log.threadName }}]  &nbsp;<span style="color: #00CCCC">{{ log.loggerName }}</span>&nbsp;:  &nbsp;{{ log.message }}  </li>  </ul>  </el-card>  </div> </template> <script setup lang="ts"> import dayjs from "dayjs" import { listLogLink, LogLinkResultVO, LogLinkVo } from "@/disco/api/modules/common"; import { reactive, ref } from "vue"; import { ResultData } from "@/disco/api/interface"; const form = reactive<LogLinkVo>({  appName: 'shipber',  trackId: '',  startTime: '',  endTime: '' }) let topSpacing = 120; let logHeight = ref((window.innerHeight-topSpacing)+'px'); window.addEventListener('resize', () => {  logHeight.value = (window.innerHeight-topSpacing)+'px'; }); let logs = reactive<{ list: LogLinkResultVO[] }>({ list: [] }) const wrap = ref(false); let formartTime = (time: any) => {  return dayjs(parseInt(time)).format("YYYY-MM-DD HH:mm:ss.SSS"); }  let showLog = () => {  let params = { ...form }  params.startTime = params.startTime && dayjs(params.startTime).valueOf() + '';  params.endTime = params.startTime && dayjs(params.endTime).valueOf() + '';  listLogLink(params).then((res: ResultData<LogLinkResultVO[]>) => {  if (res && res.success) {  logs.list = res.result;  }  }); } </script> <style scoped lang="scss"> .log-container {  display: flex;  flex-direction: column;   .box-card {  margin-top: -5px;  flex: 1;  width: 100%;  margin-bottom: 0;   .text {  font-family: consolas, Courier, "MS Courier New", monospace;  font-size: 12px !important;  line-height: 18px !important;  white-space: pre;  }   .item {  padding: 0px 0;  }   .wrap {  white-space: pre-wrap;  word-wrap: break-word;  padding: 2px 0;  }   .spacing {  margin: 0 10px;  }  } }  .el-form-item {  margin-bottom: 0px; } </style> |

**展示效果示例**

