### 1. 背景

##### 1.1 功能概述

基于 Java 中DelayQueue类构建的服务，旨在高效管理具有延迟执行需求的任务。通过提供丰富的操作接口，包括对延迟任务的添加、查询、删除等操作，为任务的延迟管理提供了便利的解决方案。同时，通过全面的日志记录功能，对所有操作过程进行详细记录，方便对系统运行状态进行监控与问题排查。

##### 1.2 例如

在处理具有时间延迟特性的业务场景中，可以利用DelayQueue - Util服务来管理这些任务。

### 2. 使用说明

### **2.1 添加任务到延迟队列**

* ****方法****：addTask
  + ****入参****：
    - ****task****：任务对象DelayedTask，它包含了丰富的任务属性信息，如delay（延迟时间，单位为毫秒，用于明确任务需要延迟多长时间后执行，该属性在计算任务在队列中的执行顺序方面起到关键作用）、startTime（任务计划开始执行的时间戳，单位为毫秒，结合当前时间和延迟时间可以判断任务是否到期）、taskId（任务的唯一标识符，在查询、删除等操作中，可依据此属性定位特定任务）、status（任务当前状态，如 "pending", "running", "completed", "failed" 等，用于监控任务执行进度）、parameters（任务执行所需参数，可以是 JSON 字符串或其他格式，为任务执行提供必要输入数据）、result（任务执行后的结果，用于存储任务执行后的输出或处理结果）、taskName（任务名称，用于描述任务的用途或类型，方便在日志记录和业务逻辑中进行识别和描述）。
  + ****出参****：添加成功返回true，失败返回false
  + ****描述****：执行此操作时，首先会检查延迟队列是否已经完成初始化。若未初始化，则抛出IllegalStateException异常；接着检查任务对象是否为null，若是则记录错误日志并返回false。若任务不为null且队列已初始化，则尝试将任务放入队列，若在放入过程中出现异常，则记录异常日志，中断当前线程，并返回false；若成功放入队列，则记录添加成功的日志并返回true。

### **2.2 尝试添加任务到延迟队列（队列满时返回false）**

* ****方法****：offer
  + ****入参****：
    - ****task****：任务对象DelayedTask。
  + ****出参****：添加成功返回true，队列满返回false
  + ****描述****：此方法首先检查延迟队列是否已初始化以及任务是否为null。若队列未初始化，则抛出IllegalStateException异常；若任务为null，则记录错误日志并返回false。若队列已初始化且任务不为null，则调用delayQueue.offer方法尝试添加任务，根据该方法的返回值判断添加是否成功，成功则记录日志，失败则记录相应的日志并返回false。

### **2.3 获取并移除最早过期的任务**

* ****方法****：take
  + ****出参****：返回最早过期的任务对象DelayedTask，如果队列为空则阻塞等待
  + ****描述****：在执行此操作前，首先检查延迟队列是否已初始化，若未初始化则抛出IllegalStateException异常。若队列已初始化，则调用delayQueue.take方法获取任务，若在获取过程中线程被中断，则记录异常日志，恢复线程中断状态，并返回null；若成功获取任务，则记录获取成功的日志并返回该任务。

### **2.4 立即获取最早过期的任务（不阻塞）**

* ****方法****：pollNow
  + ****出参****：返回最早过期的任务对象DelayedTask，如果队列为空或所有元素都未到期则返回null
  + ****描述****：首先检查延迟队列是否已初始化，若未初始化则抛出IllegalStateException异常。若队列已初始化，则直接调用delayQueue.poll方法获取任务，若获取到任务则记录日志，若队列为空则直接返回null。

### **2.5 尝试在指定时间内获取队列头部任务**

* ****方法****：pollWithTimeout
  + ****入参****：
    - ****timeout****：等待时间（毫秒）。
  + ****出参****：在指定时间内获取到的任务对象DelayedTask，超时或队列为空则返回null
  + ****描述****：首先检查延迟队列是否已初始化，若未初始化则抛出RuntimeException异常。若队列已初始化，则根据传入的timeout值（若为null则设为 0）调用delayQueue.poll方法获取任务，若在获取过程中线程被中断，则记录异常日志，恢复线程中断状态，并返回null；若成功获取任务，则记录获取成功的日志并返回该任务。

### **2.6 查看但不删除队列头部任务**

* ****方法****：peek
  + ****出参****：返回队列头部的任务对象DelayedTask，如果头部元素未到期则返回null
  + ****描述****：首先检查延迟队列是否已初始化，若未初始化则抛出IllegalStateException异常。若队列已初始化，则调用delayQueue.peek方法获取头部任务，若获取到任务则记录日志，若头部元素不存在或未到期则返回null。

### **2.7 查询并返回延迟队列中的所有任务**

* ****方法****：findAllTasks
  + ****出参****：返回包含所有任务的列表List<DelayedTask>
  + ****描述****：执行此操作时，首先检查延迟队列是否已初始化，若未初始化则抛出IllegalStateException异常。若队列已初始化，则创建一个ArrayList，将延迟队列中的所有任务添加到该列表中，并将其转换为不可变列表后返回。

### **2.8 根据任务标题模糊匹配延迟队列中的任务**

* ****方法****：searchTasksByTaskName
  + ****入参****：
    - ****query****：用于匹配任务标题的查询字符串。
  + ****出参****：返回匹配查询字符串的任务列表List<DelayedTask>
  + ****描述****：首先检查延迟队列是否已初始化，若未初始化则抛出IllegalStateException异常。若队列已初始化，则判断查询字符串是否为null或空，若是则直接返回一个空的不可变列表；接着判断队列是否为空，若是则也返回空列表。若队列不为空且查询字符串有效，则遍历队列中的任务，将任务标题包含查询字符串的任务添加到一个新的列表中，并将该列表转换为不可变列表后返回。

### **2.9 根据任务 ID 查询延迟队列中的特定任务**

* ****方法****：getTaskByTaskId
  + ****入参****：
    - ****taskId****：要查询的任务 ID。
  + ****出参****：返回匹配任务 ID 的任务对象DelayedTask，如果未找到则返回null
  + ****描述****：首先检查延迟队列是否已初始化，若未初始化则抛出IllegalStateException异常。若队列已初始化，则检查任务 ID 是否为null或空，若是则直接返回null；接着检查队列是否为空，若是则返回null。若队列不为空且任务 ID 有效，则使用迭代器遍历队列中的任务，找到任务 ID 匹配的任务并返回，若遍历完队列仍未找到匹配的任务则返回null。

### **2.10 清除延迟队列中的所有任务**

* ****方法****：clearAllTasks
  + ****出参****：清除操作成功返回true，失败返回false
  + ****描述****：首先检查延迟队列是否已初始化，若未初始化则抛出IllegalStateException异常。若队列已初始化，则调用delayQueue.clear方法清除所有任务，若清除过程中出现异常，则记录异常日志并返回false；若清除成功则记录日志并返回true。

### **2.11 从延迟队列中删除指定的任务**

* ****方法****：remove
  + ****入参****：
    - ****o****：要删除的任务对象DelayedTask。
  + ****出参****：删除操作成功返回true，失败返回false
  + ****描述****：首先检查延迟队列是否已初始化，若未初始化则抛出IllegalStateException异常。若队列已初始化，则检查要删除的任务对象是否为null，若是则记录警告日志并返回false；若任务对象不为null，则调用delayQueue.remove方法删除任务，根据该方法的返回值判断删除是否成功，成功则记录删除成功的日志并返回true，失败则返回false。

### **2.12 获取延迟队列中所有元素的数量**

* ****方法****：size
  + ****出参****：返回队列中的元素数量Integer
  + ****描述****：首先检查延迟队列是否已初始化，若未初始化则抛出IllegalStateException异常。若队列已初始化，则直接调用delayQueue.size方法获取队列元素数量并返回。

## **3. 日志记录**

在DelayQueue - Util服务的运行过程中，所有操作都会进行详细的日志记录，包括成功添加任务、添加任务失败、获取任务、删除任务等操作。具体的日志信息涵盖操作的方法名、操作的对象（如添加的任务、删除的任务等）以及操作的结果（如成功或失败的提示）等，通过这些详细的日志信息，可以方便地对系统的运行状态进行监控与分析。

## **4. 任务实体介绍**

### **4.1 概述**

DelayedTask是用于表示延迟任务的 Java 类，它实现了java.util.concurrent.Delayed接口，这使得DelayedTask的实例可以被顺利放入延迟队列中，从而实现按照指定的延迟时间进行任务执行的功能。

### **4.2 属性**

#### **4.2.1 延迟时间（Delay）**

* ****名称****：delay
* ****类型****：Long
* ****描述****：明确任务需要延迟多长时间后执行，单位为毫秒，此属性在确定任务在延迟队列中的执行顺序方面起到关键作用。

#### **4.2.2 任务开始时间（Start Time）**

* ****名称****：startTime
* ****类型****：Long
* ****描述****：表示任务计划开始执行的时间戳，单位为毫秒，通过与当前时间和延迟时间的结合，可以准确判断任务是否已经到期。

#### **4.2.3 任务 ID（Task ID）**

* ****名称****：taskId
* ****类型****：String
* ****描述****：作为任务的唯一标识符，在对特定任务进行查询、删除等操作时，可以依据此属性快速定位到目标任务。

#### **4.2.3 任务状态（Status）**

* ****名称****：status
* ****类型****：String
* ****描述****：用于表示任务的当前状态，例如 "pending", "running", "completed", "failed" 等，通过此属性可以方便地对任务的执行进度进行监控。

#### **4.2.4 任务参数（Parameters）**

* ****名称****：parameters
* ****类型****：String
* ****描述****：在任务执行时，此属性提供了任务执行所需的参数，可以是 JSON 字符串或其他格式，为任务的执行提供必要的输入数据。

#### **4.2.5 任务结果（Result）**

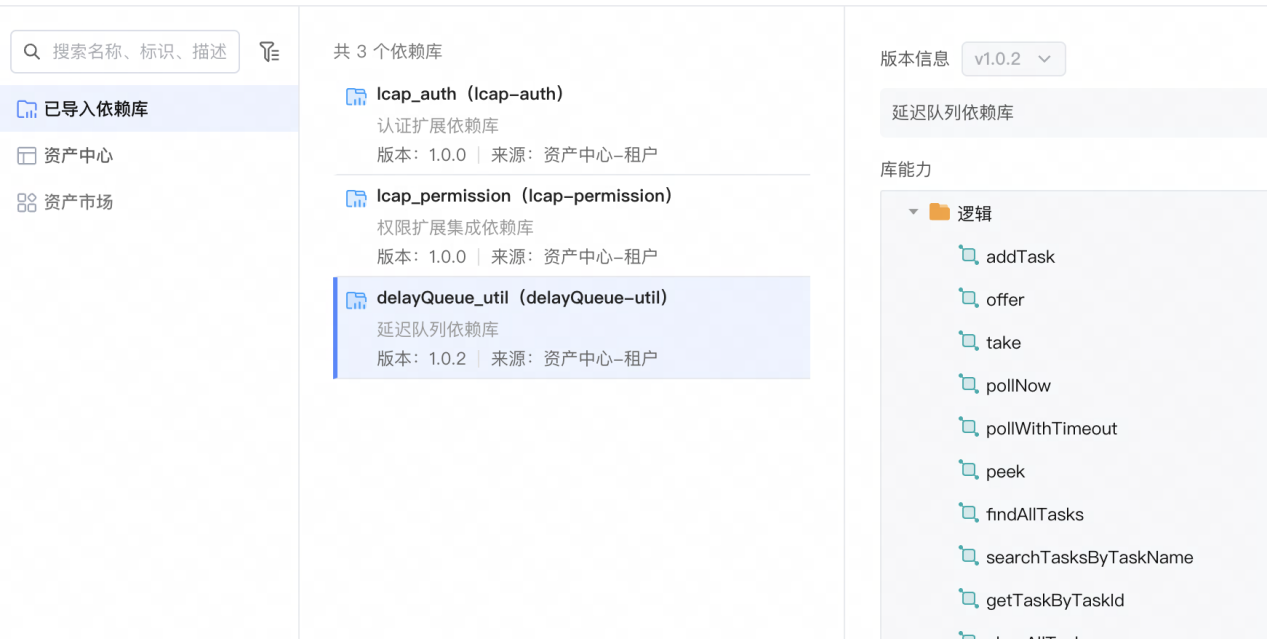
* ****名称****：result
* ****类型****：String
* ****描述****：用于存储任务执行后的结果，可以是返回值、状态信息或其他相关数据，方便对任务执行后的输出或处理结果进行获取与分析。

#### **4.2.6 任务名称（Task Name）**

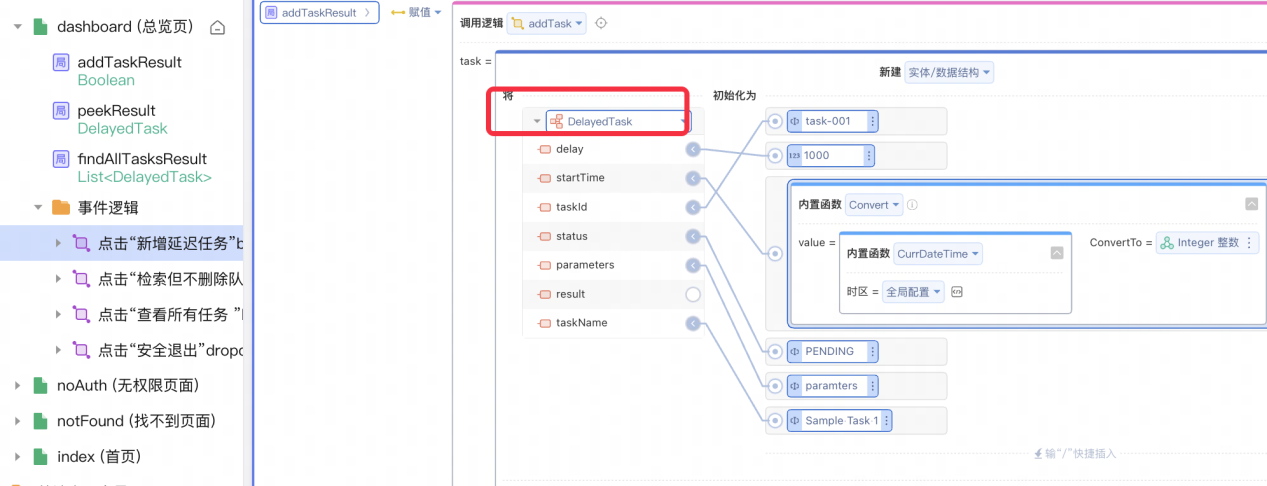
* ****名称****：taskName
* ****类型****：String
* ****描述****：用于描述任务的用途或类型，在日志记录和业务逻辑中，此属性可以方便对任务进行识别和描述。

### 使用案例

3.1引入依赖库



3.2调用方法传入参数



3.4IDE页面内调用显示（测试）



3.5查看结果

