# **MongoDbService 类功能分析文档**

## **一、类概述**

MongoDbService 类负责与 MongoDB 数据库进行交互，提供数据的存储、查询和统计功能。它封装了 MongoDB 的操作，使得其他部分的代码可以方便地进行数据访问。

## **二、功能函数分析**

### **1. 构造函数**

**public MongoDbService(string connectionString, string databaseName = "PlcDatabase", string collectionName = "PlcData")**

· **功能**：初始化 MongoDB 服务，连接到指定的数据库和集合。

· **实现思路**：

· 使用 MongoClient 创建数据库连接，传入连接字符串。

· 获取指定的数据库和集合，确保后续操作可以在正确的上下文中进行。

· 调用 CreateIndexes 方法创建必要的索引，以提高查询性能。

### **2. 创建索引**

**private async void CreateIndexes()**

· **功能**：创建必要的索引以提高查询性能。

· **实现思路**：

· 使用 Builders<PlcData>.IndexKeys 定义索引字段（如时间戳）。

· 使用 CreateOneAsync 方法异步创建索引，确保在数据插入时索引可用。

· 通过索引优化查询性能，特别是在按时间戳排序或过滤时。

### **3. 保存单条数据**

**public async Task SaveDataAsync(PlcData data)**

· **功能**：将单条 PLC 数据保存到 MongoDB。

· **实现思路**：

· 使用 InsertOneAsync 方法将数据插入集合。

· 在插入过程中，捕获可能的异常（如网络问题、数据库不可用等）。

· 如果发生异常，抛出自定义异常 MongoDbServiceException，以便调用者能够处理错误。

### **4. 批量保存数据**

**public async Task SaveManyAsync(IEnumerable<PlcData> dataList)**

· **功能**：批量保存 PLC 数据。

· **实现思路**：

· 使用 InsertManyAsync 方法将多个数据插入集合。

· 通过异步操作提高性能，减少数据库的访问次数。

· 捕获异常并抛出自定义异常，确保调用者能够处理错误。

### **5. 获取最新数据**

**public async Task<List<PlcData>> GetLatestDataAsync(int limit = 100)**

· **功能**：获取最新的 PLC 数据记录。

· **实现思路**：

· 使用 Find 方法查找所有文档，传入 \_ => true 以匹配所有记录。

· 使用 Sort 方法按时间戳降序排序，确保最新的数据在前。

· 使用 Limit 方法限制返回的记录数量，避免一次性加载过多数据。

· 返回查询结果列表，供调用者使用。

### **6. 按时间范围查询数据**

**public async Task<List<PlcData>> GetDataByTimeRangeAsync(DateTime startTime, DateTime endTime)**

· **功能**：根据时间范围查询 PLC 数据。

· **实现思路**：

· 使用 Builders<PlcData>.Filter 构建查询条件，确保只获取在指定时间范围内的数据。

· 使用 Find 方法执行查询，并返回符合条件的数据列表。

· 捕获异常并抛出自定义异常，确保调用者能够处理错误。

### **7. 删除指定时间之前的数据**

**public async Task<long> DeleteDataBeforeAsync(DateTime beforeTime)**

· **功能**：删除指定时间之前的所有数据。

· **实现思路**：

· 使用 Builders<PlcData>.Filter 构建过滤条件，确保只删除在指定时间之前的数据。

· 使用 DeleteManyAsync 方法执行删除操作，并返回删除的记录数。

· 捕获异常并抛出自定义异常，确保调用者能够处理错误。

### **8. 获取数据统计信息**

**public async Task<PlcDataStatistics> GetStatisticsAsync(DateTime startTime, DateTime endTime)**

· **功能**：获取指定时间范围内的统计信息。

· **实现思路**：

· 使用 GetDataByTimeRangeAsync 方法获取指定时间范围内的数据。

· 计算记录数量、平均值、最大值和最小值，使用 LINQ 进行统计。

· 返回统计信息对象 PlcDataStatistics，供调用者使用。

## **三、实现思路总结**

· **封装性**：将 MongoDB 的操作封装在服务类中，提供简单易用的接口，减少其他模块对数据库操作的直接依赖。

· **异步操作**：使用异步方法提高性能，避免阻塞 UI 线程，确保应用程序的响应性。

· **错误处理**：通过自定义异常类处理错误，提供更清晰的错误信息，便于调试和维护。

· **索引管理**：在初始化时创建索引，提高查询效率，特别是在数据量较大时。

· **数据模型**：使用 PlcData 和 PlcDataStatistics 数据模型，确保数据结构清晰，便于后续操作和维护。

# **PlcCommunicationService 类功能分析文档**

## **一、类概述**

PlcCommunicationService 类负责与 PLC（可编程逻辑控制器）进行通信，提供连接、数据读取、数据写入和周期性读取等功能。它封装了与 PLC 的通信细节，使得其他部分的代码可以方便地进行数据交互。

## **二、功能函数分析**

### **1. 构造函数**

**public PlcCommunicationService(string ipAddress, int port)**

· **功能**：初始化 PLC 通信服务，设置连接参数。

· **实现思路**：

· 创建 ModbusTcpNet 实例，传入 PLC 的 IP 地址和端口号。

· 设置接收和连接超时时间，以确保通信的稳定性。

· 初始化一个 ConcurrentDictionary 用于存储周期性读取的定时器。

### **2. 连接到 PLC**

**public async Task<bool> ConnectAsync()**

· **功能**：异步连接到 PLC。

· **实现思路**：

· 使用锁机制确保线程安全，避免多个线程同时尝试连接。

· 调用 ConnectServerAsync 方法尝试连接 PLC。

· 根据连接结果更新 \_isConnected 状态，并触发连接状态变化事件。

· 捕获异常并触发错误发生事件，返回连接状态。

### **3. 断开与 PLC 的连接**

**public void Disconnect()**

· **功能**：断开与 PLC 的连接。

· **实现思路**：

· 使用锁机制确保线程安全，避免多个线程同时尝试断开连接。

· 调用 ConnectClose 方法关闭连接。

· 更新 \_isConnected 状态，并触发连接状态变化事件。

· 捕获异常并触发错误发生事件。

### **4. 读取 16 位整数**

**public async Task<short?> ReadInt16Async(string address)**

· **功能**：异步读取指定地址的 16 位整数。

· **实现思路**：

· 检查是否已连接到 PLC，未连接则返回 null。

· 调用 ReadInt16Async 方法读取数据。

· 如果读取成功，触发数据变化事件并返回读取的值。

· 捕获异常并触发错误发生事件，返回 null。

### **5. 启动周期性读取**

**public void StartCyclicRead(string address, TimeSpan interval, Action<short?> callback)**

· **功能**：启动指定地址的周期性读取。

· **实现思路**：

· 创建一个 Timer，定期调用读取方法。

· 在定时器的回调中，异步读取数据并调用传入的回调函数处理读取的值。

· 将定时器存储在 \_cyclicReadTimers 字典中，以便后续管理。

### **6. 停止周期性读取**

**public void StopCyclicRead(string address)**

· **功能**：停止指定地址的周期性读取。

· **实现思路**：

· 从 \_cyclicReadTimers 字典中移除指定地址的定时器。

· 释放定时器资源，确保不再执行读取操作。

### **7. 事件处理**

**protected virtual void OnDataChanged(PlcDataChangedEventArgs e)**

**protected virtual void OnConnectionStateChanged(PlcConnectionEventArgs e)**

**protected virtual void OnErrorOccurred(PlcErrorEventArgs e)**

· **功能**：触发相应的事件。

· **实现思路**：

· 使用事件委托通知订阅者数据变化、连接状态变化或错误发生。

· 通过事件参数传递相关信息，便于调用者处理。

### **8. 资源释放**

**public void Dispose()**

· **功能**：释放资源。

· **实现思路**：

· 遍历并释放所有周期性读取的定时器。

· 断开与 PLC 的连接，释放相关资源。

· 调用 Dispose 方法释放 ModbusTcpNet 实例。

## **三、实现思路总结**

· **封装性**：将 PLC 的通信操作封装在服务类中，提供简单易用的接口，减少其他模块对通信细节的依赖。

· **异步操作**：使用异步方法提高性能，避免阻塞 UI 线程，确保应用程序的响应性。

· **错误处理**：通过事件机制处理错误，提供更清晰的错误信息，便于调试和维护。

· **周期性读取**：支持周期性读取功能，便于实时监控 PLC 数据。

· **线程安全**：使用锁机制确保多线程环境下的安全性，避免数据竞争和状态不一致。

# **WPF 系统函数功能表**

## **1. PlcCommunicationService**

### **函数列表**

· **ConnectAsync()**

· **功能**：异步连接到 PLC。

· **实现思路**：使用 ModbusTcpNet 的 ConnectServerAsync 方法尝试建立连接，更新连接状态并触发连接状态变化事件。

· **Disconnect()**

· **功能**：断开与 PLC 的连接。

· **实现思路**：调用 ConnectClose 方法关闭连接，更新连接状态并触发连接状态变化事件。

· **ReadInt16Async(string address)**

· **功能**：异步读取指定地址的 16 位整数。

· **实现思路**：检查连接状态，调用 ReadInt16Async 方法读取数据，返回读取的值或 null。

· **StartCyclicRead(string address, TimeSpan interval, Action<short?> callback)**

· **功能**：启动指定地址的周期性读取。

· **实现思路**：创建一个定时器，定期调用读取方法并执行回调函数处理读取的值。

· **StopCyclicRead(string address)**

· **功能**：停止指定地址的周期性读取。

· **实现思路**：从定时器字典中移除指定地址的定时器，释放资源。

· **OnDataChanged(PlcDataChangedEventArgs e)**

· **功能**：触发数据变化事件。

· **实现思路**：使用事件委托通知订阅者数据已更新。

· **OnConnectionStateChanged(PlcConnectionEventArgs e)**

· **功能**：触发连接状态变化事件。

· **实现思路**：使用事件委托通知订阅者连接状态已更改。

· **OnErrorOccurred(PlcErrorEventArgs e)**

· **功能**：触发错误发生事件。

· **实现思路**：使用事件委托通知订阅者发生了错误。

· **Dispose()**

· **功能**：释放资源。

· **实现思路**：停止所有定时器，断开与 PLC 的连接，释放相关资源。

---

## **2. MongoDbService**

### **函数列表**

· **MongoDbService(string connectionString, string databaseName, string collectionName)**

· **功能**：构造函数，初始化 MongoDB 服务。

· **实现思路**：使用 MongoClient 创建数据库连接，获取指定的数据库和集合，并调用 CreateIndexes 方法创建索引。

· **CreateIndexes()**

· **功能**：创建必要的索引以提高查询性能。

· **实现思路**：使用 IndexKeys 定义索引字段（如时间戳），异步创建索引。

· **SaveDataAsync(PlcData data)**

· **功能**：保存单条 PLC 数据到 MongoDB。

· **实现思路**：使用 InsertOneAsync 方法将数据插入集合，捕获异常并抛出自定义异常。

· **SaveManyAsync(IEnumerable<PlcData> dataList)**

· **功能**：批量保存 PLC 数据。

· **实现思路**：使用 InsertManyAsync 方法将多个数据插入集合，捕获异常并抛出自定义异常。

· **GetLatestDataAsync(int limit)**

· **功能**：获取最新的 PLC 数据记录。

· **实现思路**：使用 Find 方法查找所有文档，按时间戳降序排序并限制返回数量。

· **GetDataByTimeRangeAsync(DateTime startTime, DateTime endTime)**

· **功能**：根据时间范围查询 PLC 数据。

· **实现思路**：使用 Filter 构建查询条件，返回符合条件的数据列表。

· **DeleteDataBeforeAsync(DateTime beforeTime)**

· **功能**：删除指定时间之前的所有数据。

· **实现思路**：使用 DeleteManyAsync 方法删除符合条件的文档，返回删除的记录数。

· **GetStatisticsAsync(DateTime startTime, DateTime endTime)**

· **功能**：获取指定时间范围内的统计信息。

· **实现思路**：查询指定时间范围内的数据，计算记录数量、平均值、最大值和最小值，返回统计信息对象。

---

## **3. MainViewModel**

### **函数列表**

· **MainViewModel(PlcCommunicationService plcService, MongoDbService mongoDbService)**

· **功能**：构造函数，初始化服务和定时器。

· **实现思路**：注入 PLC 和 MongoDB 服务，注册事件处理器，初始化定时器。

· **ConnectAsync()**

· **功能**：连接到 PLC 的命令，处理连接状态和消息。

· **实现思路**：调用 PLC 服务的连接方法，更新连接状态，初始化周期性读取，启动定时器。

· **LoadHistoricalDataAsync()**

· **功能**：加载历史数据并更新 UI。

· **实现思路**：从 MongoDB 获取最新数据，清空当前历史数据集合并添加新数据。

· **DatabaseSaveTimer\_Tick(object? sender, EventArgs e)**

· **功能**：定时器事件处理，保存数据到数据库。

· **实现思路**：创建要保存的数据对象，异步保存到 MongoDB，处理异常。

· **OnConnectionStateChanged(object? sender, PlcConnectionEventArgs e)**

· **功能**：处理连接状态变化事件。

· **实现思路**：更新连接状态和消息。

· **OnErrorOccurred(object? sender, PlcErrorEventArgs e)**

· **功能**：处理错误事件，更新错误消息。

· **实现思路**：更新连接消息以反映错误信息。

· **Dispose()**

· **功能**：释放资源，确保不再使用已释放的对象。

· **实现思路**：停止定时器，断开与 PLC 的连接，标记为已释放。

---

## **4. MainWindow.xaml**

### **函数列表**

· **MainWindow()**

· **功能**：窗口构造函数，初始化窗口组件。

· **实现思路**：设置窗口的基本属性和布局。

· **InitializeComponent()**

· **功能**：初始化窗口的组件和布局。

· **实现思路**：自动生成的代码，设置 XAML 中定义的 UI 元素。

· **OnLoaded()**

· **功能**：窗口加载时的事件处理。

· **实现思路**：执行窗口加载时需要的初始化操作。

· **OnClosed()**

· **功能**：窗口关闭时的事件处理。

· **实现思路**：执行窗口关闭时需要的清理操作。