# 一：NLog的概念

NLog是一个非常强大的日志记录库，它可以用于.NET和.NET Core应用程序。NLog可以帮助你记录应用程序运行过程中的各种信息，包括错误、警告、调试信息等。这些信息对于诊断问题、优化性能和理解应用程序的行为非常有用。
以下是如何更好地使用NLog的一些建议：

1. 理解日志级别：NLog支持多种日志级别，包括Trace、Debug、Info、Warn、Error和Fatal。你应该根据日志的重要性和详细程度来选择合适的日志级别。例如，对于重要的错误，你应该使用Error或Fatal级别。对于调试信息，你应该使用Debug或Trace级别。

2. 使用结构化日志：NLog支持结构化日志，这意味着你可以在日志消息中包含键值对。这对于后期分析日志非常有用。例如，你可以这样写日志：

```
logger.Info("User {UserId} logged in at {LoginTime}", userId, DateTime.Now);
```

在这个例子中，{UserId}和{LoginTime}是结构化日志的键，userId和DateTime.Now是对应的值。

3. 配置多个目标：NLog允许你将日志发送到多个目标，包括文件、控制台、数据库、网络等。你应该根据你的需求来配置合适的目标。例如，对于开发环境，你可能想要将日志输出到控制台。对于生产环境，你可能想要将日志写入文件或发送到日志服务。

4. 使用异步日志：NLog支持异步日志，这可以避免日志记录阻塞你的应用程序。你可以在NLog配置文件中启用异步日志，如下所示：

```
<nlog>
  <targets async="true">
    <!-- your targets here -->
  </targets>
  <!-- your rules here -->
</nlog>
```

5. 使用布局渲染器：NLog支持多种布局渲染器，这可以帮助你生成丰富的日志消息。例如，你可以使用${date}渲染器来添加当前日期，使用${exception}渲染器来添加异常信息。

# 二：NLog详细使用方法

当然可以。以下是一个NLog的完整配置文件示例，该配置文件定义了两个目标（target）：一个发送到Elasticsearch，另一个发送到控制台。这个配置文件还定义了两个规则（rule），分别将日志发送到这两个目标。

```
<?xml version="1.0" encoding="utf-8" ?>
<nlog xmlns="http://www.nlog-project.org/schemas/NLog.xsd"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      autoReload="true"
      throwExceptions="false"
      internalLogLevel="Off" internalLogFile="c:\temp\nlog-internal.log">

  <!-- 定义目标 -->
  <targets>
    <!-- Elasticsearch目标 -->
    <target xsi:type="ElasticSearch" name="elastic" uri="http://localhost:9200" index="logstash-${date:format=yyyy.MM.dd}" documentType="logevent" includeAllProperties="true" layout="${message}">
    </target>

    <!-- 控制台目标 -->
    <target xsi:type="Console" name="console" layout="${date:format=HH\:mm\:ss} ${logger} ${message}" />
  </targets>

  <!-- 定义规则 -->
  <rules>
    <!-- 将所有的日志发送到Elasticsearch -->
    <logger name="*" minlevel="Info" writeTo="elastic" />

    <!-- 将所有的日志发送到控制台 -->
    <logger name="*" minlevel="Info" writeTo="console" />
  </rules>
</nlog>
```

在这个配置文件中，我们首先定义了两个目标。ElasticSearch目标将日志发送到运行在本地的Elasticsearch实例，而Console目标将日志输出到控制台。

然后，我们定义了两个规则。这两个规则都将所有的日志（级别为Info或更高）发送到对应的目标。

# 每一个节点的详细概念

以下是NLog配置文件中各个节点的详细解释：

```
<?xml version="1.0" encoding="utf-8" ?>
<nlog xmlns="http://www.nlog-project.org/schemas/NLog.xsd"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      autoReload="true"
      throwExceptions="false"
      internalLogLevel="Off" internalLogFile="c:\temp\nlog-internal.log">
```

- autoReload：这个属性决定了当NLog配置文件被修改后，是否自动重新加载配置。如果设置为true，NLog将在配置文件被修改后自动重新加载配置。
- throwExceptions：这个属性决定了当NLog遇到错误时，是否抛出异常。如果设置为false，NLog将在遇到错误时不抛出异常，而是尝试继续运行。
- internalLogLevel和internalLogFile：这两个属性用于配置NLog的内部日志。internalLogLevel决定了内部日志的级别，internalLogFile决定了内部日志的文件路径。

```
<targets>
    <target xsi:type="ElasticSearch" name="elastic" uri="http://localhost:9200" index="logstash-${date:format=yyyy.MM.dd}" documentType="logevent" includeAllProperties="true" layout="${message}">
    </target>
    <target xsi:type="Console" name="console" layout="${date:format=HH\:mm\:ss} ${logger} ${message}" />
</targets>
```

- targets：这个节点包含了所有的目标（target）。每个目标都是一个target节点。
- target：这个节点定义了一个目标。它的xsi:type属性决定了目标的类型，name属性决定了目标的名称。其他的属性则取决于目标的类型。例如，ElasticSearch目标有uri，index，documentType，includeAllProperties和layout等属性。

```
<rules>
    <logger name="*" minlevel="Info" writeTo="elastic" />
    <logger name="*" minlevel="Info" writeTo="console" />
</rules>
```

- rules：这个节点包含了所有的规则（rule）。每个规则都是一个logger节点。
- logger：这个节点定义了一个规则。它的name属性决定了这个规则应用于哪些日志，minlevel属性决定了这个规则应用于哪些级别的日志，writeTo属性决定了这个规则将日志发送到哪个目标。

```
internalLogLevel="Off"是NLog配置文件中的一个设置，它用于控制NLog的内部日志级别。

```

NLog的内部日志是NLog自身的日志系统，用于记录NLog在运行过程中的信息，包括错误、警告和其他诊断信息。这些信息对于调试NLog配置和解决NLog问题非常有用。

internalLogLevel属性用于设置这个内部日志的级别。它可以设置为以下几个值：

- Off：关闭内部日志，不记录任何信息。
- Fatal：只记录致命错误。
- Error：记录所有错误，包括致命错误。
- Warn：记录所有警告和错误。
- Info：记录一般信息、警告和错误。
- Debug：记录所有详细信息，包括一般信息、警告和错误。
- Trace：记录所有跟踪信息和详细信息，包括一般信息、警告和错误。

因此，internalLogLevel="Off"的意思是关闭NLog的内部日志，不记录任何信息。这通常在你确定你的NLog配置没有问题，且不需要额外的诊断信息时使用。

在NLog的配置文件中，target节点定义了一个日志目标，它的属性决定了这个目标的行为。以下是一些常见的target属性：

- xsi:type：这个属性决定了目标的类型。NLog支持多种类型的目标，包括文件（File）、控制台（Console）、数据库（Database）、邮件（Mail）、网络（Network）等。你需要根据你的需求来选择合适的目标类型。

- name：这个属性决定了目标的名称。这个名称在NLog的配置文件中必须是唯一的。你可以在rules节点中使用这个名称，来决定哪些日志应该发送到这个目标。

- layout：这个属性决定了日志消息的格式。你可以使用NLog的布局渲染器来生成丰富的日志消息。例如，${message}渲染器会输出日志消息，${date}渲染器会输出当前日期。

以下是一个ElasticSearch目标的例子，它包含了一些特定于ElasticSearch目标的属性：

```
<target xsi:type="ElasticSearch" name="elastic" uri="http://localhost:9200" index="logstash-${date:format=yyyy.MM.dd}" documentType="logevent" includeAllProperties="true" layout="${message}">
</target>
```

- uri：这个属性决定了Elasticsearch实例的地址。你需要将这个地址设置为你的Elasticsearch实例的地址。

- index：这个属性决定了日志消息应该发送到哪个Elasticsearch索引。你可以使用NLog的布局渲染器来动态生成索引名称。在这个例子中，我们使用了${date:format=yyyy.MM.dd}渲染器来生成索引名称，这样每天的日志都会发送到一个新的索引。

- documentType：这个属性决定了日志消息的文档类型。这个类型在Elasticsearch中是一个重要的概念，它决定了文档的结构。

- includeAllProperties：这个属性决定了是否应该将所有的日志属性包含在日志消息中。如果设置为true，所有的日志属性都会被发送到Elasticsearch。

## 三：内置布局渲染器

NLog支持许多内置的布局渲染器，以下是一些常用的内置布局渲染器及其详细说明：

- ${message}：日志事件的消息。这是日志事件的主要内容。

- ${level}：日志事件的级别。这表示日志事件的严重性，如Info、Warn、Error等。

- ${logger}：生成日志事件的logger的名称。这通常是生成日志事件的类的全名。

- ${exception}：日志事件的异常信息。如果日志事件包含一个异常，这将是异常的详细信息。

- ${date}：日志事件的日期和时间。这是日志事件发生的时间。

- ${callsite}：生成日志事件的代码位置。这是一个包含类名和方法名的字符串。

- ${longdate}：日志事件的长日期和时间格式。这是一个更详细的时间戳，包含日期和时间。

- ${machinename}：运行应用程序的机器的名称。这是生成日志事件的计算机的名称。

- ${processid}：运行应用程序的进程的ID。这是生成日志事件的进程的唯一标识符。

- ${threadid}：生成日志事件的线程的ID。这是生成日志事件的线程的唯一标识符。

- ${threadname}：生成日志事件的线程的名称。如果线程有一个名称，这将是它的名称。

- ${windows-identity}：当前Windows用户的身份。这是运行应用程序的Windows用户的名称。

- ${event-properties}：日志事件的自定义属性。这是一个字典，包含您在写日志时添加的任何自定义属性。

- ${gdc}：全局诊断上下文（Global Diagnostic Context）。这是一个全局的字典，您可以在其中存储任何信息。

- ${mdc}：映射诊断上下文（Mapped Diagnostic Context）。这是一个与当前线程关联的字典，您可以在其中存储任何信息。

- ${ndc}：嵌套诊断上下文（Nested Diagnostic Context）。这是一个与当前线程关联的堆栈，您可以在其中存储任何信息。

以上只是一部分内置布局渲染器，NLog还支持许多其他的内置布局渲染器。您可以查看[NLog的文档](https://nlog-project.org/config/?tab=layout-renderers)来获取完整的列表。