使用 XPath 的情况：

- **数据结构**：
  结构化数据，数据以一种清晰的层级结构存储数据，而且这种层级结构的顺序很重要。例如，HTML 文档中，`<div>` 内部的 `<p>` 标签通常会紧跟在 `<div>` 之后。XPath 可以利用这种顺序来定位特定的元素。
- **复杂查询**：
  需要根据元素的属性、文本内容或其在文档结构中的位置来选择特定元素的情况。例如，选择具有特定 `class` 属性的 `<div>` 内部的所有 `<span>` 元素。
- **文档类型**：
  处理 HTML 或 XML 文档，这些文档通常会以一种可扩展的方式来表示。例如，选择所有具有特定标签名的元素，或选择具有特定属性值的元素。

```
# 示例：选择 class 为 "product" 的 div 内部的所有 span 元素
//div[@class="product"]//span
```

- **结构稳定**：
  结构变化较小的情况，通常是一次性的，或需要频繁对不同结构进行调整的情况。
- **性能要求**：
  对性能要求较高的情况，特别是大规模文档。通过精确定位目标节点，可能会比使用正则表达式或字符串操作更高效。

使用 lxml 解析 HTML

```
# 示例：使用 lxml 解析 HTML
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 现在可以使用 XPath 表达式来查询 tree 对象
# 例如，选择所有 h1 标签
h1_elements = tree.xpath('//h1')

# 打印找到的 h1 标签的文本内容
for h1 in h1_elements:
    print(h1.text)
```

使用 XPath 和 requests

```
# 示例：使用 XPath 和 requests
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 使用 XPath 选择所有 h1 标签
h1_elements = tree.xpath('//h1')

# 打印找到的 h1 标签的文本内容
for h1 in h1_elements:
    print(h1.text)
```

处理 lxml 解析错误

```
# 示例：处理 lxml 解析错误
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 尝试解析 HTML 内容，处理可能的错误
try:
    tree = etree.fromstring(html_content, parser)
    # 现在可以使用 XPath 表达式来查询 tree 对象
    # 例如，选择所有 h1 标签
    h1_elements = tree.xpath('//h1')

    # 打印找到的 h1 标签的文本内容
    for h1 in h1_elements:
        print(h1.text)
except etree.ParseError as e:
    print("解析错误：", e)
```

使用 XPath 选择特定属性

```
# 示例：使用 XPath 选择特定属性
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择所有 a 标签的 href 属性
hrefs = tree.xpath('//a/@href')

# 打印找到的 href 属性
for href in hrefs:
    print(href)
```

使用 XPath 选择文本内容

```
# 示例：使用 XPath 选择文本内容
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择所有 p 标签的文本内容
paragraphs = tree.xpath('//p/text()')

# 打印找到的文本内容
for paragraph in paragraphs:
    print(paragraph)
```

使用 XPath 选择特定文本内容

```
# 示例：使用 XPath 选择特定文本内容
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择包含特定文本的 p 标签
specific_paragraphs = tree.xpath('//p[contains(text(), "特定文本")]')

# 打印找到的文本内容
for paragraph in specific_paragraphs:
    print(paragraph.text)
```

使用 XPath 选择多个属性

```
# 示例：使用 XPath 选择多个属性
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择所有 a 标签的 href 和 title 属性
links = tree.xpath('//a/@href | //a/@title')

# 打印找到的 href 和 title 属性
for link in links:
    print(link)
```

使用 XPath 选择子元素

```
# 示例：使用 XPath 选择子元素
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择所有 div 标签内的 p 标签
paragraphs = tree.xpath('//div/p')

# 打印找到的 p 标签的文本内容
for paragraph in paragraphs:
    print(paragraph.text)
```

使用 XPath 选择父元素

```
# 示例：使用 XPath 选择父元素
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择包含特定 p 标签的 div 标签
divs = tree.xpath('//p[contains(text(), "特定文本")]/parent::div')

# 打印找到的 div 标签的 class 属性
for div in divs:
    print(div.attrib['class'])
```

使用 XPath 选择兄弟元素

```
# 示例：使用 XPath 选择兄弟元素
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择包含特定 p 标签的兄弟 p 标签
siblings = tree.xpath('//p[contains(text(), "特定文本")]/following-sibling::p')

# 打印找到的兄弟 p 标签的文本内容
for sibling in siblings:
    print(sibling.text)
```

使用 XPath 选择第一个元素

```
# 示例：使用 XPath 选择第一个元素
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择第一个 p 标签
first_paragraph = tree.xpath('//p[1]')

# 打印找到的第一个 p 标签的文本内容
for paragraph in first_paragraph:
    print(paragraph.text)
```

使用 XPath 选择最后一个元素

```
# 示例：使用 XPath 选择最后一个元素
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择最后一个 p 标签
last_paragraph = tree.xpath('//p[last()]')

# 打印找到的最后一个 p 标签的文本内容
for paragraph in last_paragraph:
    print(paragraph.text)
```

使用 XPath 选择第 n 个元素

```
# 示例：使用 XPath 选择第 n 个元素
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择第 2 个 p 标签
second_paragraph = tree.xpath('//p[2]')

# 打印找到的第 2 个 p 标签的文本内容
for paragraph in second_paragraph:
    print(paragraph.text)
```

使用 XPath 选择属性值大于特定值的元素

```
# 示例：使用 XPath 选择属性值大于特定值的元素
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择 data-value 属性值大于 100 的 div 标签
divs = tree.xpath('//div[@data-value > 100]')

# 打印找到的 div 标签的 data-value 属性
for div in divs:
    print(div.attrib['data-value'])
```

使用 XPath 选择属性值小于特定值的元素

```
# 示例：使用 XPath 选择属性值小于特定值的元素
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择 data-value 属性值小于 50 的 div 标签
divs = tree.xpath('//div[@data-value < 50]')

# 打印找到的 div 标签的 data-value 属性
for div in divs:
    print(div.attrib['data-value'])
```

使用 XPath 选择属性值等于特定值的元素

```
# 示例：使用 XPath 选择属性值等于特定值的元素
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择 data-value 属性值等于 100 的 div 标签
divs = tree.xpath('//div[@data-value = 100]')

# 打印找到的 div 标签的 data-value 属性
for div in divs:
    print(div.attrib['data-value'])
```

使用 XPath 选择属性值不等于特定值的元素

```
# 示例：使用 XPath 选择属性值不等于特定值的元素
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择 data-value 属性值不等于 100 的 div 标签
divs = tree.xpath('//div[@data-value != 100]')

# 打印找到的 div 标签的 data-value 属性
for div in divs:
    print(div.attrib['data-value'])
```

使用 XPath 选择属性值包含特定子字符串的元素

```
# 示例：使用 XPath 选择属性值包含特定子字符串的元素
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择 data-value 属性值包含 "abc" 的 div 标签
divs = tree.xpath('//div[contains(@data-value, "abc")]')

# 打印找到的 div 标签的 data-value 属性
for div in divs:
    print(div.attrib['data-value'])
```

使用 XPath 选择属性值不包含特定子字符串的元素

```
# 示例：使用 XPath 选择属性值不包含特定子字符串的元素
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择 data-value 属性值不包含 "abc" 的 div 标签
divs = tree.xpath('//div[not(contains(@data-value, "abc"))]')

# 打印找到的 div 标签的 data-value 属性
for div in divs:
    print(div.attrib['data-value'])
```

使用 XPath 选择属性值以特定字符开头的元素

```
# 示例：使用 XPath 选择属性值以特定字符开头的元素
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择 data-value 属性值以 "abc" 开头的 div 标签
divs = tree.xpath('//div[starts-with(@data-value, "abc")]')

# 打印找到的 div 标签的 data-value 属性
for div in divs:
    print(div.attrib['data-value'])
```

使用 XPath 选择属性值以特定字符结尾的元素

```
# 示例：使用 XPath 选择属性值以特定字符结尾的元素
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择 data-value 属性值以 "abc" 结尾的 div 标签
divs = tree.xpath('//div[ends-with(@data-value, "abc")]')

# 打印找到的 div 标签的 data-value 属性
for div in divs:
    print(div.attrib['data-value'])
```

使用 XPath 选择属性值长度大于特定值的元素

```
# 示例：使用 XPath 选择属性值长度大于特定值的元素
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择 data-value 属性值长度大于 10 的 div 标签
divs = tree.xpath('//div[string-length(@data-value) > 10]')

# 打印找到的 div 标签的 data-value 属性
for div in divs:
    print(div.attrib['data-value'])
```

使用 XPath 选择属性值长度小于特定值的元素

```
# 示例：使用 XPath 选择属性值长度小于特定值的元素
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择 data-value 属性值长度小于 5 的 div 标签
divs = tree.xpath('//div[string-length(@data-value) < 5]')

# 打印找到的 div 标签的 data-value 属性
for div in divs:
    print(div.attrib['data-value'])
```

使用 XPath 选择属性值长度等于特定值的元素

```
# 示例：使用 XPath 选择属性值长度等于特定值的元素
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择 data-value 属性值长度等于 10 的 div 标签
divs = tree.xpath('//div[string-length(@data-value) = 10]')

# 打印找到的 div 标签的 data-value 属性
for div in divs:
    print(div.attrib['data-value'])
```

使用 XPath 选择属性值长度不等于特定值的元素

```
# 示例：使用 XPath 选择属性值长度不等于特定值的元素
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择 data-value 属性值长度不等于 10 的 div 标签
divs = tree.xpath('//div[string-length(@data-value) != 10]')

# 打印找到的 div 标签的 data-value 属性
for div in divs:
    print(div.attrib['data-value'])
```

使用 XPath 选择属性值为空的元素

```
# 示例：使用 XPath 选择属性值为空的元素
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择 data-value 属性值为空的 div 标签
divs = tree.xpath('//div[@data-value = ""]')

# 打印找到的 div 标签的 data-value 属性
for div in divs:
    print(div.attrib['data-value'])
```

使用 XPath 选择属性值不为空的元素

```
# 示例：使用 XPath 选择属性值不为空的元素
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择 data-value 属性值不为空的 div 标签
divs = tree.xpath('//div[@data-value != ""]')

# 打印找到的 div 标签的 data-value 属性
for div in divs:
    print(div.attrib['data-value'])
```

使用 XPath 选择属性值为数字的元素

```
# 示例：使用 XPath 选择属性值为数字的元素
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择 data-value 属性值为数字的 div 标签
divs = tree.xpath('//div[@data-value = number(@data-value)]')

# 打印找到的 div 标签的 data-value 属性
for div in divs:
    print(div.attrib['data-value'])
```

使用 XPath 选择属性值为非数字的元素

```
# 示例：使用 XPath 选择属性值为非数字的元素
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择 data-value 属性值为非数字的 div 标签
divs = tree.xpath('//div[@data-value != number(@data-value)]')

# 打印找到的 div 标签的 data-value 属性
for div in divs:
    print(div.attrib['data-value'])
```

使用 XPath 选择属性值为日期的元素

```
# 示例：使用 XPath 选择属性值为日期的元素
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择 data-value 属性值为日期的 div 标签
divs = tree.xpath('//div[@data-value = date(@data-value)]')

# 打印找到的 div 标签的 data-value 属性
for div in divs:
    print(div.attrib['data-value'])
```

使用 XPath 选择属性值为非日期的元素

```
# 示例：使用 XPath 选择属性值为非日期的元素
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择 data-value 属性值为非日期的 div 标签
divs = tree.xpath('//div[@data-value != date(@data-value)]')

# 打印找到的 div 标签的 data-value 属性
for div in divs:
    print(div.attrib['data-value'])
```

使用 XPath 选择属性值为布尔值的元素

```
# 示例：使用 XPath 选择属性值为布尔值的元素
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择 data-value 属性值为布尔值的 div 标签
divs = tree.xpath('//div[@data-value = boolean(@data-value)]')

# 打印找到的 div 标签的 data-value 属性
for div in divs:
    print(div.attrib['data-value'])
```

使用 XPath 选择属性值为非布尔值的元素

```
# 示例：使用 XPath 选择属性值为非布尔值的元素
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择 data-value 属性值为非布尔值的 div 标签
divs = tree.xpath('//div[@data-value != boolean(@data-value)]')

# 打印找到的 div 标签的 data-value 属性
for div in divs:
    print(div.attrib['data-value'])
```

使用 XPath 选择属性值为特定类型的元素

```
# 示例：使用 XPath 选择属性值为特定类型的元素
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择 data-value 属性值为特定类型的 div 标签
divs = tree.xpath('//div[@data-value = type(@data-value)]')

# 打印找到的 div 标签的 data-value 属性
for div in divs:
    print(div.attrib['data-value'])
```

使用 XPath 选择属性值为非特定类型的元素

```
# 示例：使用 XPath 选择属性值为非特定类型的元素
import requests
from lxml import etree

# 发送 HTTP 请求获取网页内容
response = requests.get('https://example.com')
html_content = response.text

# 创建 HTML 解析器对象
parser = etree.HTMLParser()

# 解析 HTML 内容
tree = etree.fromstring(html_content, parser)

# 选择 data-value 属性值为非特定类型的 div 标签
divs = tree.xpath('//div[@data-value != type(@data-value)
```