XPath Query

XPathQuery Class

Anonymous · 09/09/2026

XPathQuery Class

The `XPathQuery` class is a powerful tool for extracting data from XML documents using XPath expressions. It provides a convenient interface to query XML with support for namespaces, parameterized queries, and complex data structures.

Basic Usage

```
use Derafu\Xml\XPathQuery;

// Initialize with XML string or DOMDocument.
$query = new XPathQuery($xmlString);

// Get a single value.
$value = $query->getValue('/root/element');

// Get multiple values.
$values = $query->getValues('/root/items/item');

// Get DOM nodes.
$nodes = $query->getNodes('/root/items/item');

// Get structured data.
$data = $query->get('/root');
```

Working with Namespaces

```
// Initialize with namespace support.
$namespaces = [
    'ns' => 'http://example.com/ns',
    'xs' => 'http://www.w3.org/2001/XMLSchema'
];

$query = new XPathQuery($xmlString, $namespaces);

// Use namespaces in queries.
$result = $query->get('//ns:Element/xs:Type');
```

Parameterized Queries

The `XPathQuery` class supports parameterized queries, making it safer and easier to build dynamic XPath expressions:

```
// Query with parameters.
$params = [
    'id' => '123',
    'type' => 'product'
];

$result = $query->get('//item[@id=:id and @type=:type]', $params);
```

This automatically handles escaping of values, including proper handling of values containing quotes.

Context Nodes

You can limit the scope of your query by providing a context node:

```
// Get a context node.
$contextNode = $query->getNodes('//section')->item(0);

// Query within that context.
$result = $query->get('./item', [], $contextNode);
```

Return Value Handling

The `get()` method is intelligent about what it returns:

1. `null` if no matching nodes are found.
2. A string if a single node with no children is found.
3. An array if multiple nodes are found.
4. A structured array if nodes with child elements are found.

Example of Structured Results

For XML like:

```
<root>
  <person>
    <name>John</name>
    <age>30</age>
```

```
  </person>
  <person>
    <name>Jane</name>
    <age>25</age>
  </person>
</root>
```

The query `$query->get('/root/person')` would return:

```
[
    [
        'name' => 'John',
        'age' => '30'
    ],
    [
        'name' => 'Jane',
        'age' => '25'
    ]
]
```

Special Features

Using Without Namespaces

If you need to query XML with namespaces but don't want to specify them all, you can use the class without registering namespaces:

```
$query = new XPathQuery($xmlString); // No namespaces registered.

// This will match any element named 'Element' regardless of its namespace.
$result = $query->get('//Element');
```

Value Quoting

The class handles the quoting of values in XPath expressions automatically, even when values contain both single and double quotes:

```
$params = ['complex' => 'Value with "double" and \'single\' quotes'];
$result = $query->get('//element[text()=:complex]', $params);
```

Error Handling

The class provides clear error messages when there are issues with the XML document or XPath expressions:

```
try {
    $result = $query->get('//invalid[xpath');
} catch (InvalidArgumentException $e) {
    // Handle the error.
}
```

Last updated on 09/09/2026