

Первым шагом в понимании XPath является понимание того, что полученные результаты в большой степени определяются текущим контекстным узлом. Контекстный узел можно считать своеобразным знаком "ты находишься здесь", от которого можно двигаться в различных направлениях согласно выражению XPath. Например, рассмотрим простую таблицу стилей в листинге 14.

### Листинг 14. Демонстрация контекста

```
<xsl:stylesheet version="1.0"
```

```
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
```

```
xmlns="http://www.w3.org/TR/xhtml1/strict">
```

```
<xsl:template match="/">
```

```
<xsl:copy-of select="." />
```

```
</xsl:template>
```

```
</xsl:stylesheet>
```

Здесь есть новый элемент - `copy-of`. Там, где `value-of` выводит содержимое элемента, `copy-of` делает именно то, о чем говорит его название, и выводит фактический узел, на который ссылается атрибут `select`.

В данном случае в атрибуте `select` содержится одно из простейших возможных выражений XPath. Одиночная точка (`.`) – это ссылка на данный контекстный узел, как в файловой системе, когда вы хотите сослаться на "текущую директорию, вне зависимости от того, что это за директория." Поэтому если вы выполните это преобразование, то получите результат, как в листинге 15.

### Листинг 15. Простейшее преобразование

<recipes>

<recipe recipeld="1">

<name>Gush'gosh</name>

<ingredients>

<ingredient><qty>1</qty><unit>pound</unit>

<food>hamburger </food></ingredient>

<ingredient><qty>1</qty><unit>pound</unit>

<food>elbow macaroni</food></ingredient>

<ingredient><qty>2</qty><unit>cups</unit>

<food>brown sugar</food></ingredient>

<ingredient><qty>1</qty><unit>bag</unit>

<food>chopped onions</food></ingredient>

<ingredient><qty>1</qty><unit>teaspoon</unit>

<food>dried dill</food></ingredient>

</ingredients>

<instructions>

<instruction>Brown the hamburger.</instruction>

<instruction>Add onions and cook until transparent.</instruction>

<instruction>Add brown sugar and dill.</instruction>

```
<instruction>Cook and drain pasta.</instruction>
```

```
<instruction>Combine meat and pasta.</instruction>
```

```
</instructions>
```

```
</recipe>
```

```
...
```

```
</recipes>
```

Вы видите, что это просто повторение того же документа; это потому, что контекстный узел, как указано атрибутом шаблона `match`, - это корень документа, или `/`. Если вы измените контекстный узел, вывод изменится. Например, вы можете установить контекстный узел на первый рецепт (см. листинг 16).

### Листинг 16. Перемещение контекстного узла

```
<xsl:stylesheet version="1.0"
```

```
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
```

```
xmlns="http://www.w3.org/TR/xhtml1/strict">
```

```
<xsl:template match="/recipes/recipe">
```

```
<xsl:copy-of select="." />
```

```
</xsl:template>
```

```
</xsl:stylesheet>
```

Теперь, запустив преобразование, вы увидите разницу (см. листинг 17).

### Листинг 17. Результаты

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<recipe recipeld="1">
```

```
<name>Gush'gosh</name>
```

```
<ingredients>
```

```
<ingredient><qty>1</qty><unit>pound</unit>
```

<food>hamburger</food></ingredient>

<ingredient><qty>1</qty><unit>pound</unit>

<food>elbow macaroni</food></ingredient>

<ingredient><qty>2</qty><unit>cups</unit>

<food>brown sugar</food></ingredient>

<ingredient><qty>1</qty><unit>bag</unit>

<food>chopped onions</food></ingredient>

<ingredient><qty>1</qty><unit>teaspoon</unit>

<food>dried dill</food></ingredient>

</ingredients>

<instructions>

<instruction>Brown the hamburger.</instruction>

<instruction>Add onions and cook until transparent.</instruction>

<instruction>Add brown sugar and dill.</instruction>

<instruction>Cook and drain pasta.</instruction>

<instruction>Combine meat and pasta.</instruction>

</instructions>

</recipe>

<recipe recipeld="2">

<name>A balanced breakfast</name>

<ingredients>

<ingredient><qty>1</qty><unit>cup</unit>

<food>cereal</food></ingredient>

<ingredient><qty>1</qty><unit>glass</unit>

<food>orange juice</food></ingredient>

<ingredient><qty>1</qty><unit>cup</unit>

<food>milk</food></ingredient>

```
<ingredient><qty>2</qty><unit>slices</unit>
```

```
<food>toast</food></ingredient>
```

```
</ingredients>
```

```
<instructions>
```

```
<instruction>Combine cereal and milk in bowl.</instruction>
```

```
<instruction>Add all ingredients to table.</instruction>
```

```
</instructions>
```

```
</recipe>
```

Так как мы переместили контекстный узел, вывод изменился. Это важно, когда вы хотите выбрать узел относительно контекстного узла. Например, вы можете выбрать только заголовок рецепта (см. листинг 18).

## Листинг 18. Выбор только заголовка

```
<xsl:stylesheet version="1.0"
```

```
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
```

```
xmlns="http://www.w3.org/TR/xhtml1/strict">
```

```
<xsl:template match="/recipes/recipe">
```

```
<xsl:copy-of select="./name" />
```

```
</xsl:template>
```

```
</xsl:stylesheet>
```

Здесь вы выбираете элемент `name`, расположенный на один уровень ниже, чем текущий контекстный узел, поэтому результат будет как в листинге 19.

### Листинг 19. Результаты

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<name>Gush'gosh</name>
```

```
<name>A balanced breakfast</name>
```

Мы изучим задание узлов по номеру позже, а пока отметим, что мы можем переместить контекстный узел во второй рецепт (см. листинг 20).

### Листинг 20. Еще одно перемещение контекстного узла

```
<xsl:stylesheet version="1.0"
```

```
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
```

```
xmlns="http://www.w3.org/TR/xhtml1/strict">
```

```
<xsl:template match="/recipes/recipe">
```

```
</xsl:template>
```

```
<xsl:template match="/recipes/recipe[2]">
```

```
<xsl:copy-of select="./name" />
```

```
</xsl:template>
```

```
</xsl:stylesheet>
```

(Не обращайте внимания на еще один шаблон, он нужен, чтобы не появлялся текст другого рецепта.)

Мы можем видеть, что результирующее преобразование показывает это изменение контекста (см. листинг 21).

### Листинг 21. Результаты

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<name>A balanced breakfast</name>
```

### Для чего нужны оси

Теперь, когда мы знаем, откуда мы начинаем, было бы полезно узнать, куда можно двигаться. До сих пор мы использовали очень простые выражения XPath, похожие на иерархию в файловой системе, однако XPath предоставляет гораздо больше возможностей. Например, до сих пор вы выбирали только дочерние элементы, однако вы можете также искать родителей конкретного узла, а также его потомков или предков.

Для начала давайте поговорим о системе обозначений. Мы использовали упрощенную, укороченную форму запроса потомков. В "длинной форме" выражения задаются как `child::name` вместо `./name`

В обоих случаях вы задаете любой узел, который является потомком контекстного узла, а также является элементом `name`. Вы также можете связать их вместе, как в предыдущем случае, используя `/child::recipes/child::recipe` вместо `/recipes/recipe`.

### Потомки

Возможно, вы недоумеваете, зачем мучиться с длинной формой, если короткая настолько проще. Это не было бы нужно, если бы единственное, что вы могли бы делать - это выбор потомков. Однако эту нотацию также можно использовать для выбора других отношений. Например, выражение XPath `descendant::instruction` выбирает все элементы `instruction`, которые являются потомками контекстного узла, а не только дочерние элементы. Кроме того, можно комбинировать инструкции. Например, вы можете выбрать все инструкции второго рецепта:  
`/recipes/recipe[2]/descendant::instruction.`

Одной из разновидностей оси потомков является ось потомок-или-сам-элемент, которая выбирает заданный узел из всех потомков, а также ищет в самом контекстном узле. Например, выражение `descendant-or-self::instructions` выбирает все узлы `instruction` в контекстном узле и ниже его. Обычным сокращением для этой оси является двойная косая черта, `//`. Это означает, что выражения: `/recipes/recipe[2]//instructions` и `//instructions` выбирают все инструкции для второго рецепта и все инструкции в документе, соответственно. Этот второй пример очень широко применяется, он удобен, когда вы хотите просто выбрать все элементы определенного типа в документе.

## Атрибуты

Еще одна общая задача, с которой вы столкнетесь - это необходимость выбрать атрибут для конкретного элемента. Например, вы, возможно, захотите выбрать `recipeId` для конкретного рецепта. Выражение `/recipes/recipe/attribute::recipeId` выбирает атрибут `recipeId` для всех элементов `recipe`. Эта ось также имеет сокращенную форму: это же выражение можно записать так: `/recipes/recipe/@recipeId`.

## Родители

Все, что мы видели до этого, переносило нас *вниз* по дереву иерархии, но также существует возможность пойти *вверх*, выбрав предка конкретного узла. Предположим, что контекстным узлом была одна из инструкций, а вы хотите вывести `recipeId` для текущего рецепта. Вы можете сделать это следующим образом: `./parent::node()/parent::node()/@recipeId`.

Это выражение начинается в текущем узле - `instruction` -, а затем двигается к предку этого узла - элементу `instructions`- а затем к предку ТОГО элемента - `recipe` -, а затем к соответствующему атрибуту. Возможно, вам лучше знакома сокращенная форма: `../../@recipeId`.

Теперь давайте посмотрим, как задаются некоторые условия.

### Более продвинутые возможности XPath

В большинстве случаев уже рассмотренных приемов достаточно. Однако нередко встречаются ситуации, требующие большей избирательности. Данный раздел показывает, как использовать предикаты для выбора узлов, основанных на конкретных критериях, и дает представление о некоторых функциях, встроенных в XPath.