Recognizable query patterns

Query patterns are described as a CSS3 selector pattern (*http://www.w3.org/TR/css3-selectors*). The following table shows all available query patterns of the query tool.

Pattern	Meaning
E	An element of type E.
E[foo]	An E element with a “foo” attribute.
E[foo=bar]	An E element whose “foo” attribute value exactly equals “bar”.
E[foo^=bar]	An E element whose “foo” attribute value begins exactly with the string “bar”.
E[foo$=bar]	An E element whose “foo” attribute value ends exactly with the string “bar”.
E[foo*=bar]	An E element whose “foo” attribute value contains the substring string “bar”.
E:empty	An E element that has no children.
#myid	An element with an ID equals “myid”.
E F	An F element descendant of an E element.
E > F	An F element child of an E element.

Query by Type

A pattern allows a template to query elements from a given type name, for example:

```
class
Retrieve all classes from a selected package scope.
```

A Type name is generated from the $element.elementType property.

Query by Attribute

A pattern allows a template to query elements from given attribute name and value. When a pattern is used as an expression to match an element, the attribute pattern must be considered to match the element and that element must have an attribute that matches the attribute represented by the attribute pattern.

[attr]
: Matches an element with an attr attribute regardless of the value of the attribute.

[attr=value]
: Matches an element with an attr attribute whose value is exactly val.

The Query tool matches against case sensitivity of attribute names and values in a pattern. A *VTL* property syntax (Camel case without space) will be used for the attribute names. You can use spaces in writing values.

When the value of an attribute is NamedElement, the element’s name will be used as a default value. An asterisk (*) can be used to match any element type when combined with the other patterns, for example:

[name=foo]
Retrieve any element whose name attribute is exactly foo. Use an asterisk to match any element.

actor[name=foo]
Retrieve all actor elements whose name attribute is exactly foo.

actor[name=]
Retrieve all actor elements whose name attribute is empty.

class[owner=foo]
Retrieve class elements whose owner attribute name is exactly foo.

class[owner=foo][name=bar]
Retrieve class elements whose owner attribute name is exactly foo and name attribute is exactly bar.

Query by Substring Matching Attribute

This topic provides some additional attribute patterns for matching substrings in the value of an attribute:

[attr^=val]
Matches an element with an attr attribute whose value begins with the prefix val. If val is an empty string, the query returns nothing.

[attr$=val]
Matches an element with an attr attribute whose value ends with the suffix val. If val is an empty string, the query returns nothing.

[attr*=val]
Matches an element with an attr attribute whose value contains at least one instance of the substring val. If val is an empty string, the query returns nothing.

The Query tool matches against case sensitivity of attribute names and values in a pattern. A *VTL* property syntax (Camel case without space) will be used for attribute name. You can use spaces in writing values.

When value of attribute is NamedElement, the element name will be used as default value, for example:

```
*[name^=foo]
Retrieve any element whose name attributes begins with foo. Use an asterisk to match any element.
actor[name$=foo]
Retrieve all actor elements whose name attribute is ends with foo.
class[qualifiedName*=nomagic]
Retrieve class elements whose qualifiedName attribute contains string nomagic.
```

Query Element without Children

You can use :empty to query an element that has no children at all. If an element does not any children, its “Owned Element” attribute value has a zero size of items, for example:

```
package:empty
Retrieve any package whose ownedElement attribute is empty.
package[name=foo]:empty
Retrieve any package whose name attribute is exactly foo and ownedElement attribute is empty.
```

Query by Element ID

A pattern allows a template to query elements from a given element’s ID. An element’s ID is unique in a project, for example:

```
#17_0_3_8e9027a_1325220298609_477646
Retrieve an element whose ID is exactly 17_0_3_8e9027a_1325220298609_477646.
```

An element’s ID is generated from the $element.elementID property.

Query by Descendent

A pattern allows a template to query an element that is the descendant of another element, for example, an Actor that is contained within a Package. A descendant query is one or more white-spaces that separate(s) two sequences of patterns.

You can use an asterisk (*) to match any element type when combined with the other patterns, for example:

```
package class
Retrieve all classes that are the descendants of a package element.
package foo
 - class bar1
 - class bar2
 - class bar3
The classes bar1, bar2, and bar3 are the results.
package class *[name=foo]
Retrieve any element that (i) has a name value exactly foo and (ii) is inside a class that is inside a package.
package * property
Retrieve a property element that is the grandchild or a later descendant of a package element.
```

Query by Child Element

A pattern allows a template to query elements that are the children of another element (describes a parent-child relationship between two elements). An optional white-space around a greater sign can be ignored.

You can use an asterisk (*) to match any element type when combined with the other patterns, for example:

```
package > class
Retrieve all classes that are children of a package.
package foo
 - class bar1
 - class bar2
 - class bar3
The classes bar1 and bar2 are the results.
```

Additional Conditions for Query Patterns

When you are writing queries, be sure you observe the following conditions.

- A Query pattern cannot contain only spaces, for example:

```
$query.get('') or $query.get(' ')
An invalid query pattern.
```

- An element type cannot contain [,], :, *, >, #, and white space.
- Characters #, >, and white-space are the end statements except in an attribute name or value, for example:

```
$query.get("class#17_0_3_8e9027a") equivalent to
$query.get("class #17_0_3_8e9027a")
Retrieve an element's ID 17_0_3_8e9027a that is a grandchild or a later descendant of a class element.
$query.get("class>property") equivalent to
$query.get("class > property")
Retrieve a property element that is the child of a class element.
$query.get("class property")
Retrieve a property element that is a grandchild or a later descendant of a class element.
```

An end statement is required to separate an element and its child, or grandchild, for example:

```
$query.get("class[attr] property")
Retrieve a property element that is a grandchild or a later descendant of a class element that contains an attr
attribute.
$query.get("class[attr]property")
An invalid query pattern.
```

- Characters = , ^=, $=, and *= are keywords and all strings except] that are written next to these keywords mean the attribute values.
- Attribute names are mandatory.
- An attribute name cannot contain white-space, =, ^=, $=, *=, and], for example:

```
$query.get("*[ name =bean]") is equivalent to
$query.get("*[name=bean]")
Retrieve all elements whose name attribute is bean.
$query.get("*[ element type = bean ] ")
An invalid query pattern.
```

If an attribute value is not in any double quotes or single quotes, all strings, except], after a keyword mean an attribute value, for example:

```
$query.get("*[name=bean]")
 Retrieve all elements whose name attribute is bean
```

- Character # must contain at least one string.
- An ID cannot contain [, : , >, #, and white-space, for example:

```
$query.get("#17_0_3_8e9027a#17_0_3_8e9027b") is equivalent to
$query.get("#17_0_3_8e9027a #17_0_3_8e9027b")
Retrieve an element ID #17_0_3_8e9027b that is a grandchild or a later descendant of an element ID
17_0_3_8e9027a.
$query.get("#17_0_3_8e9027a > *")
Retrieve all elements that are the children of an element ID #17_0_3_8e9027a.
$query.get("#") or
$query.get("#[name=a]#17_0_3_8e9027a")
An invalid query pattern.
```

- A : must be followed by empty and ":empty" must be followed by [, #, >, or white-space, except in an attribute, for example:

```
$query.get("*:empty") or
$query.get("*:empty#17_0_3_8e9027a") or
$query.get("*:empty > class") or
$query.get("*:empty[name*=class]")
A valid query pattern but the result of some patterns will be an empty list.
$query.get("*[name=:emptyx]")
Retrieve all elements whose name attribute is :emptyx
$query.get("*:emptyx") or
$query.get("*:em")
An invalid query pattern.
```

- If an attribute value of an element is null, it is equivalent to an empty attribute value.