Search Templates

Search templates are the essential part of Structural Search and Replace feature. Like Live templates, the search templates consist of plain text and one or more *template variables*.

A valid search or replacement template represents one of the following Java constructs:

- Expression, for example `new SomeExpression()`
- Statement, or sequence of statements, for example `q = 1;`
- Class designator, for example `class Booking implements Serializable`
- Line or block comments, for example `/** Created in IntelliJ IDEA */`.
- `@Modifier` annotations.

IntelliJ IDEA provides a collection of predefined search templates, that match the various statements, expressions, classes and their members, XML and HTML constructs, and more. You can use these templates for structural search and replace, and also as a basis for creating your own search templates.

The search templates make use of the *variables*, which are the strings surrounded with `$` characters, for example `$expression$`. Symbols in the source code, `String` literals, and comments can be referred to by means of variables.

Variables in a template are subjected to certain constraints that help you refine your search and limit it to the desired matches:

- *Text constraints* are text patterns to match against. These constraints can be plain text, or regular expressions, and can contain references to symbols.
- *Number of occurrences* defines how many sequential elements (in a parameter, declaration or statement list) a variable can include and whether a variable required to be present in a pattern or not. If the number of occurrences is 1, only one symbol can match the variable. If the number of occurrences is null, it means that an element could be missing.
- *Expression constraints* apply semantic conditions to the search, for example locate the symbols that are read or written to.
- *Script constraints* are used when items to search for are more than a plain match. If you are looking for certain language constructs (for example, constructors with the specified number of parameters, or members with the specified visibility modifiers), apply constraints described as Groovy scripts.

In search templates, the following simplifications can be used:

- Method body can be omitted.
- If no access modifier is indicated, any access modifier will be honored.
- Short class names (instead of fully qualified names) are used in the templates and constraint fields.
- Using `class $Class$` as a template, results in finding anonymous classes as well.
- Templates for comments and JavaDoc should contain variables and constructs with correct comment and JavaDoc syntax.

See Also

Procedures:

- Structural Search and Replace
- Creating and Editing Search Templates

Reference:

- Structural Search and Replace Dialogs

- Structural Search and Replace. Edit Variable Dialog

Web Resources:

- Developer Community