# Guide: Using Groovy language for Cytomine

This document is a useful summary for people who don't know the Groovy language.
We list here most of the Groovy code that we use inside the Cytomine code.
**This is not all the Groovy rules, this is the main used rules inside Cytomine (non-exhaustive!!!).**

| Java way | Groovy way | Note |
|---|---|---|
| `statement;` | `statement` | ; is not mandatory at the end of a statement. |
| `return var;` | `var` | If method is not void and no return statement,<br>Groovy return the last exec statement in a function |
| `Object var;`<br>`int var;`<br>`List var;`<br><br>`int foo(int var) {`<br>`  return var;`<br>`}` | `def var`<br>`int number`<br><br>`def foo(def var) {`<br>`  var`<br>`}` | Auto typing. Java way is OK in Groovy too |
| `class User {`<br>`  private String name;`<br><br>`  public String getName() {...}`<br>`  public void setName(String name) {...}`<br><br>`}` | `class User {`<br>`  private String name;`<br><br>`  //getter/setter auto generated`<br>`}` | Getter and setter auto generated for private members. |
| | `def object = ...`<br>`object.newField = "test"` | You can add new field or new method to an object at Runtime. |
| `java.io.*`<br>`java.lang.*`<br>`java.math.BigDecimal`<br>`java.math.BigInteger`<br>`java.net.*`<br>`java.util.*`<br>`groovy.lang.*`<br>`groovy.util.*` | | Groovy auto import all these packages.<br><br>Aliases can be used in imports with "as" keyword (as in Python). |

| | | |
|---|---|---|
| `if (annotation.getTerm() != null && annotation.getTerm().getUser() != null && annotation.getTerm().getUser().getId() == 1) { ... }` | `if (annotation.getTerm()?.getUser()?.getUser() == 1) { ... }` | Safe navigation operator " ?. " |
| `String name = p.getName() != null ? p.getName() : "";` | `def name = p.getName() ?: ""` | Elvis operator " ?: ".<br><br>To assign a value or return a default value if it<br><br>is evaluated to false. |
| `try { methodMayThrowException() } catch(...) {...}` | `methodMayThrowException()` | |
| `String s = "hello " + name`<br>`String s = "hello " + user.getName();` | `String s = "hello $name"`<br>`String s = "hello ${user.getName()}` | GStrings |
| `List<String> list = new list<String>();` | `def list = []`<br>`def listNotEmpty = [1,2,3]`<br>`println listNotEmpty[0] // prints 1`<br>`println listNotEmpty[-1] // prints 3`<br><br>`// Find in list`<br>`if (x in list) { ... }` | Syntacttic sugar for lists.<br><br>Similar to Python lists. |
| `Map<Long,String> map = new Map<Long, String>();`<br><br>`String x = map.get(2);` | `def map = [:]`<br>`def mapNotEmpty = [1:"a",2:"b", 3:"c"]`<br><br>`def x = map[2] //or map.get(2)`<br><br>`def k = 'key3'`<br>`def map = [`<br>`'key1': 1,`<br>`key2: 2, // Skip quotes, seen as String`<br>`(k): 3 // Put () for variable`<br>`]`<br>`assert map[k] == map['key3'] == map.key3` | Syntactic sugar for maps. |

| | | |
|---|---|---|
| `for(String item : list) {`<br>`  doSomething(item);`<br>`}` | `list.each {`<br>`  doSomething(it);`<br>`}` | Apply function on each element of a list.<br><br>By default, the current item in each{} is it. You can change this by:<br><br>`.each {item -> ...}`<br><br>You can get the index (current position in list):<br><br>`.eachWithIndex {item, i -> ...}` |
| `List<User> users = ...`<br>`List<String> names = ...`<br><br>`for(User user : users) {`<br>`  names.add(user.getName());`<br>`}` | `def names = users.collect{it.getName())` | Build a list from anohter one. |
| `System.out.println("hello");` | `println("hello")`<br>`println "hello"` | Print. |
| +/- Java 8 lambdas | `def square = { it * it }`<br>`println square(2) // prints 2`<br><br>`def power = { x, y ->`<br>`  Math.pow(x, y)`<br>`}`<br>`println power(2, 3) // prints 8` | Closures. |
| `Boolean x = (a == b); // Equality of primitive types`<br>`x = a.equals(b); // Equality of instances` | `def x = a.is(b) // Equality of primitive types`<br>`x = (a == b) // Equality of instances` | Equals behaviour. |