Tips and tricks about creating your own placeholders

We recommend to write a little SQL script file for each placeholder you want to define yourself, like in this example

file: MY_PAT_OCCUPATION.sql

```
INSERT INTO letter.placeholders (name, type, comment, script)
VALUES ('[MY_PAT_OCCUPATION]', 'DUMMY', 'dummy', 'dummy');

UPDATE letter.placeholders
SET type = 'PATIENT_INFO',
    comment = 'The patient''s occupation',
    script =
'/**
 * [MY_PAT_OCCUPATION]
 */

PATIENT.getJob() == null ? "" :
    PATIENT.getJob();'
WHERE name = '[MY_PAT_OCCUPATION]';
```

The above script has the following advantages over a direct INSERT:

- The INSERT “Dummy” statement creates a new entry with an ID that is generated automatically.
- The UPDATE statement sets the actual values. While figuring out the correct JavaScript code you will have to make some changes. This script can be executed after every change you make, without knowing the actual ID of the placeholder.
- When exeuting the script more than once you will get a “unique key violation” error for the INSERT part, but the UPDATE part will still work fine. Therefore your changes are taken into account.

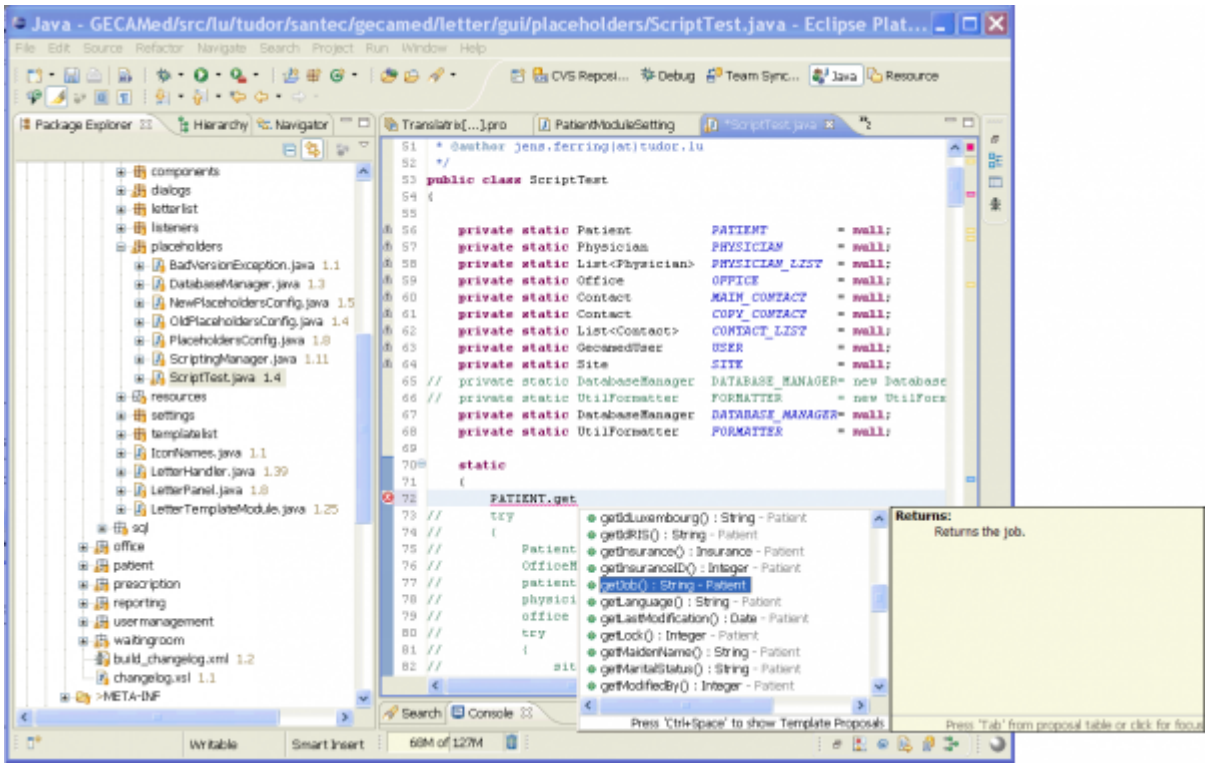
Once finished you can put all the single placeholder creation scripts into one bigger file (easier to load), or keep them apart and add a load script on top of them.

How do I know which methods can be called in the JavaScript?

Best is to look into the GECAMed sources, as they might change over time. We provide a class called “ScriptTest.java” under “/GECAMed/src/lu/tudor/santec/gecamed/letter/gui/placeholders” which defines static variables like PATIENT, PHYSICIAN, OFFICE and so on for all the objects that the JavaScript engine is able to access during placeholder processing.

Moreover, with a Java-aware IDE like Eclipse you can use the autocompleter function (Ctl-Space) to

get a listing af all known methods for a given object, like PATIENT. This makes it much easier to figure out which methods are actually available. (see below)



You can even try to build JavaScript code in the Java editor this way, provided that the corresponding JavaScript function exist.

To test if your new placeholder works, you have to put it into a letter template and try it out. If there are errors, this will be marked in the generated letter at the place where the placeholder once stood. It loks like taht: a«`ScriptError: [MY_PAT_OCCUPATION]`»

Moreover, the client log file has more information about the nature of the error, like this one:

```
sun.org.mozilla.javascript.internal.EvaluatorException: unterminated string
literal (<Unknown source>#28) in <Unknown source> at line number 28
2013-11-25 12:40:57,796
lu.tudor.santec.gecamed.letter.gui.placeholders.ScriptingManager
evalPlaceholder (line 202)
ERROR: Error in script of LetterPlaceholder [MY_PAT_OCCUPATION]
SCRIPT:
/**
 * [MY_PAT_OCCUPATION]
 */

patient.getJob() == null ? "" :
    patient.getJob();

sun.org.mozilla.javascript.internal.EcmaError: ReferenceError: "patient" is
not defined. (<Unknown source>#5) in <Unknown source> at line number 5
2013-11-25 12:41:11,343
lu.tudor.santec.gecamed.core.utils.controller.document.DocumentController
startReplacing (line 135)
```

In this case the error was that “patient” has to be written in all uper case, because this is how the constant is defined.