Ruby Script tool API

‘eval’ Method

eval(String script)

This method will evaluate a Ruby code from a string and return the result.

```
$ruby.eval("puts 'Hello World!'")
```

eval(String script, String bindingName, Object bindingObject)

This method will evaluate a Ruby code with a single binding object and specified binding name. The code will be evaluated from a string. The binding name will be used as the name for this object.

```
#foreach ($class in $Class)
$ruby.eval('"Class name is " + c.name', 'c', $class)
#end
```

eval(String script, Map bindingMap)

This method will evaluate a Ruby code with a set of binding arguments (a name and an object). The code will be evaluated from a string. The binding map consists of key-value pairs for the binding name and binding object.

```
#set ($dict = $map.createHashMap())
#set ($void = $dict.put("first", "foo"))
#set ($void = $dict.put("last", "bar"))
$ruby.eval("puts first + last", $dict)
```

Another alternative is as follows:

```
$ruby.eval("puts first + last", {"first":"foo", "last":"bar"})
```

The second code contains curly brackets; “{“ and “}” characters, which are not allowed to be used in any *RTF* template. For the *RTF* template, use the first code instead.

‘execute’ Method

execute(String filename)

This method will execute a Ruby file. The ‘filename’ parameter is a name of the Ruby file or an absolute path to the Ruby file.

After executing the Ruby file, the result of the execution will be stored in the single context for each report generation. If the Ruby file contains Ruby functions, you can recall the functions with the use of ‘eval()’ methods.

For example, File ‘String.rb’:

```
def deCamelCase(str)
return str.gsub!(/(.)([A-Z])/,'\1 \2')
end
```

Indicated below is the template code.

```
$ruby.execute("String.rb")
#foreach ($c in $Class)
$ruby.eval('deCamelCase($c.name)')
#end
```

execute(String filename, String bindingName, Object bindingObject)

This method will execute a Ruby file with a single binding object and specified binding name. The ‘filename’ parameter is a file path to the Ruby file. The binding name will be used as the name for this object.

```
File 'filename.rb'
puts c.name
```

The template code

```
#foreach ($c in $Class)
$ruby.execute("filename.rb", 'c', $c)
#end
```

execute(String filename, Map bindingMap)

This method will execute a Ruby file with a set of binding arguments (a name and an object). The ‘filename’ parameter is a file path to the Ruby file. The binding map consists of key-value pairs for the binding name and binding object.

File '*filename.rb*'

```
puts first+' '+last
```

The template code

```
#set ($dict = $map.createHashMap())
#set ($void = $dict.put("first", "foo"))
#set ($void = $dict.put("last", "bar"))
$ruby.execute("filename.rb", $dict)
```

- **Absolute Path**
  If the ‘filename’ is provided with an absolute path, Ruby tool will read the Ruby file from an absolute location such as $ruby.execute(‘*c:/mycode/readclass.rb*’).
- **Relative Path**
  If the ‘filename’ is provided with a relative path, Ruby tool will read the template from a relative location. This relative location starts from the current directory location of the template, such as $ruby.execute(‘*readclass.rb*’).