[DEPRECATED] Part 6: Python implementation

Implementation of a Python client-side resource

The PYTHON client allow, as much as the previously explained Java client, communication with the Cytomine *Web API*. This client was developed to be integrated as a library in Python app. We use it the same way than the Java client :

- It contains a Cytomine object which send HTTP requests and has methods to manipulate resources : addUser(...), editUser(...), removeUser(...), addAnnotation(...), ...
- It is mandatory to define a Model per resource.

The differences between the clients are mainly due to the respective language constraints. We will describe the way to define a model in Python for the resource *User* without the details common with the Java client and previously explained on its page.

Super-class Model : each model must extend this class

```
#Models
class Model(object):

   def __init__(self, params = None):
       if (params != None):
           self.parse(params)

   def parse(self, params = None):
       obj = json.loads(params)
       print obj
       self.__dict__ = obj

   def toJSON(self):
       return json.dumps(self.__dict__)

   def isNew(self):
       return not hasattr(self, "id")
```

Definition of a User model for the Python client

```
class User(Model):

   def __init__(self, params = None):
       super(User, self).__init__(params)
       self._callback_identifier = "user"

   def toURL(self):
       if (hasattr(self, "id")):
           return "user/%d.json" % self.id
       else:
           return "user.json"

   def __str__( self ):
       return "User : " + str(self.id)
```

Example: How to use Python client

```
[...]
cytomine = Cytomine("http://localhost:8080", "dupond", "pass")

//Add the user johndoe
user = cytomine.addUser("johndoe", "Password","John", "Doe")

//Display name/firstname of all the users
users = cytomine.getUsers()
for user in users.data():
    print "%s %s" % (user.firstname,user.lastname)

[...]
```