

```

1  #!/usr/bin/python3.5
2  # -*-coding:Utf-8 -*
3
4  from json import loads
5  from re import findall
6  from operator import itemgetter
7
8  from .JSONEncoder import JSONEncoder
9  from .PathManager import PathManager
10 from src.factories.IAFactory import IAFactory
11 from src.EvolutiveGenerator.ProcessusState import ProcessusState
12 from src.EvolutiveGenerator.event_names import *
13
14
15 class Reader:
16     """Read files
17
18     This is a static class.
19
20     Public API:
21     processusExists(processus_id)
22     getProcessusState(processus_id)
23     getIa(processus_id, ia_id)
24     getBestIa(processus_id, generation_id=None)
25     getData(processus_id)
26     """
27
28
29     @staticmethod
30     def getPath(*args, **kwargs):
31         return PathManager.getPath(*args, **kwargs, read_only=True)
32
33
34     @staticmethod
35     def readJSON(path):
36         return loads(path.read_text())
37
38
39     @staticmethod
40     def readGrading(path):
41         return tuple(loads(json_array)) for json_array in findall('\[.+\]', path.read_text())
42
43
44     @classmethod
45     def getProcessusParams(cls, processus_id):
46         path = cls.getPath(processus_id)
47         if not path.parent.exists():
48             raise ValueError("Processus {} doesn't exists.".format(processus_id))
49         if not path.exists():
50             raise ValueError("Processus {} doesn't have processus.json file.".format(processus_id))
51
52         return cls.readJSON(path)
53
54
55     @classmethod
56     def getLastGeneration(cls, processus_id, generations):
57         '''Get id of the processus' last generation, else -1'''
58         # Get first inexistant generation
59         generation_id = 0
60         while cls.getPath(processus_id, generations, generation_id).parent.exists():
61             generation_id += 1
62
63         return generation_id - 1
64
65
66     @classmethod
67     def getLastGradedGeneration(cls, processus_id, generations):
68         # Get first inexistant final_grading file's generation
69         generation_id = 1
70         while cls.getPath(processus_id, generations, generation_id, 'final_grading').exists():
71             generation_id += 1
72
73         return generation_id - 2
74
75
76     @classmethod
77     def getGenerationOf(cls, processus_id, generations, ia_id):
78         generation_id = 1
79         while True:
80             path = cls.getPath(processus_id, generations, generation_id, 'final_grading')
81             if not path.exists():
82                 raise ValueError("IA {} doesn't exist !".format(ia_id))
83             final_grading = cls.readJSON(path)
84             for score, _ia_id in final_grading:
85                 if _ia_id == ia_id:
86                     return generation_id - 1
87             generation_id += 1
88
89         raise RuntimeError
90
91
92     @classmethod
93     def getPopulation(cls, processus_id, generation_id, generations):
94         population = set()
95         for ia_file in (
96             cls.getPath(processus_id, generations, generation_id).parent
97             / ('population' if generation_id > 0 else 'initial_pop')
98         ).iterdir():
99             population.add(IAFactory.hydrate(cls.readJSON(ia_file)))

```

```

100         return population
101
102
103 @classmethod
104 def getIa(cls, processus_id, ia_id):
105     generations = cls.getProcessusParams(processus_id)['generations']
106     generation_id = cls.getGenerationOf(processus_id, generations, ia_id)
107     ia_file = cls.getPath(processus_id, generations, generation_id, ia_id)
108     return IAFactory.hydrate(cls.readJSON(ia_file)), generation_id
109
110
111 @classmethod
112 def getBestIa(cls, processus_id, generation_id = None):
113     generations = cls.getProcessusParams(processus_id)['generations']
114     if generation_id is None:
115         generation_id = generations if type(generations) is int else cls.getLastGradedGeneration(processus_id, generations)
116     grading = cls.readJSON(cls.getPath(processus_id, generations, generation_id + 1, 'final_grading'))
117     grading.sort(key=lambda c: c[0]['score'], reverse=True)
118     ia_id = grading[0][1]
119     ia_file = cls.getPath(processus_id, generations, generation_id, ia_id)
120     return IAFactory.hydrate(cls.readJSON(ia_file)), generation_id
121
122
123 @classmethod
124 def processusExists(cls, processus_id):
125     path = cls.getPath(processus_id)
126     if not path.parent.exists():
127         return False
128     return True
129
130
131 @classmethod
132 def getData(cls, processus_id):
133     generation_id = 1
134     generations = cls.getProcessusParams(processus_id)['generations']
135     data = []
136
137     while True:
138         path = cls.getPath(processus_id, generations, generation_id, 'final_grading')
139         if not path.exists():
140             break
141         final_grading = cls.readJSON(path)
142         data.append((generation_id - 1, final_grading))
143         generation_id += 1
144
145     return data
146
147
148 @classmethod
149 def getProcessusState(cls, processus_id):
150     '''
151     for state.event_name in (
152         PROCESSUS.START,
153         CREATION.START,
154         CREATION.DONE,
155         GENERATION.START,
156         GRADING.START,
157         GRADING.PROGRESS,
158         GRADING.DONE,
159         SELECTION.START,
160         SELECTION.DONE,
161         BREEDING.START,
162         BREEDING.PROGRESS,
163         BREEDING.DONE,
164         GENERATION.DONE,
165         PROCESSUS.DONE
166     )
167     '''
168
169     state = ProcessusState()
170     state.processus_id = processus_id
171     state.__dict__.update(cls.getProcessusParams(processus_id))
172
173     getPath = lambda generation_id, file_name = None: cls.getPath(
174         processus_id, state.generations, generation_id, file_name
175     )
176
177     generation_id = cls.getLastGeneration(processus_id, state.generations)
178     # If none generation folder exist
179     if generation_id == -1:
180         state.event_name = PROCESSUS.START
181         return state
182     state.generation_id = generation_id
183
184     # Get event_name
185     state.event_name = cls.readJSON(getPath(state.generation_id))['event_name']
186
187     if state.event_name in (CREATION.DONE, BREEDING.DONE, GENERATION.DONE, PROCESSUS.DONE):
188         state.population = cls.getPopulation(state.processus_id, state.generation_id, state.generations)
189     else:
190         state.population = cls.getPopulation(state.processus_id, state.generation_id - 1, state.generations)
191
192     if state.event_name in (GRADING.PROGRESS):
193         state.grading = cls.readGrading(getPath(state.generation_id, 'grading'))
194     elif state.event_name in (GRADING.DONE, SELECTION.START):
195         state.grading = cls.readJSON(getPath(state.generation_id, 'final_grading'))
196     if state.grading is not None:
197         indexed_pop = dict([(ia.id, ia) for ia in state.population])
198         state.grading = [(score, indexed_pop[ia_id]) for (score, ia_id) in state.grading]
199
200     if state.event_name in (SELECTION.DONE, BREEDING.START, BREEDING.PROGRESS):

```

```
201         state.selection = cls.readJSON(getPath(state.generation_id, 'selection'))
202
203     return state
```