|  |
| --- |
| The Bank of New York Mellon Corporation |
| MLEngine |
| Understanding MLEngine |
|  |
| **Kalia, Urvi Raveen** |
| **2/10/2017** |

|  |
| --- |
| Document helps us understand MlEngine design and its consumption. |

Table of Contents

[MLEngine WebService 2](#_Toc475111553)

[Class Diagram 3](#_Toc475111554)

[Deployment 4](#_Toc475111555)

[Configurations 4](#_Toc475111556)

[Consuming MLEngine 5](#_Toc475111557)

[MlEngine Client using API 5](#_Toc475111558)

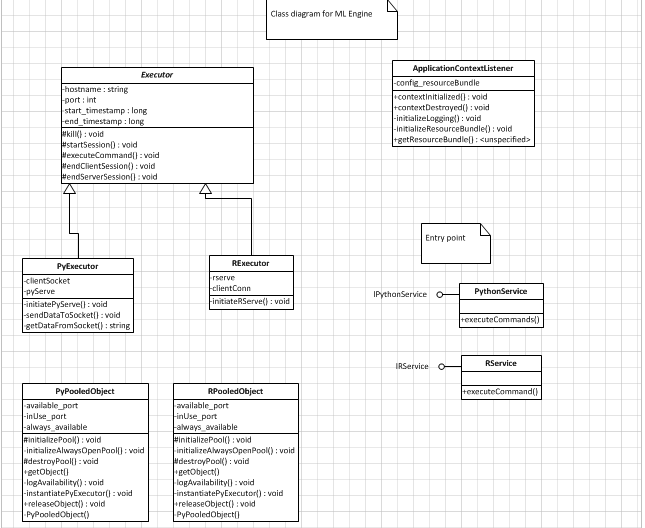
[Enhancements /Changes 6](#_Toc475111559)

[TODO 6](#_Toc475111560)

# MLEngine WebService

* MLEngine is built as Restful Web Service using CFX.
* MlEngine handles instantiation of RServe/PyServe along with their clients and manages their pooling mechanism.
* Web service has two end points.
  + For RService : http://<hostname:port>/MlEngine/services/RService.
  + For PythonService: http://<hostname:port>/MlEngine/services/PythonService
* RService
  + - Method : executeCommand
    - Type: POST
    - Input : RRequest
    - Output : RResponse
    - Consumes : JSON
    - Produces : JSON
* PythonService
  + - Method : executeCommands
    - Type: POST
    - Input : PyRequest
    - Output : PyResponse
    - Consumes : JSON
    - Produces : JSON

# Class Diagram



# Deployment

Currently MlEngine projects is not a maven project, still the process is pretty straightforward. Though deployment on the server will be a manual process as of now.

Steps to be followed:

* WAR creation: Right click on the project and say export as WAR.
* Place the created WAR file into <TOMCAT>/webapps folder.
* Restart the server.
* Can check server logs for any startup issues.
* MlEngine application is up and running.
* Can verify using tomcat manager console.

## Configurations

There are few configuration changes one might want to change. These are included in config.properties file within MlEngine project.

Parameters which are configurable are:

|  |  |
| --- | --- |
| **Property key** | **Details** |
| hostname | Hostname is used by pythonClient /RConnection to make the connection to the PyServe/RServe.Technically value of hostname specifies where pyServe/rServe are running. |
| *pythonSockServeLocation* | Specify the complete absolute path for sockServe python file. |
| *python\_start\_port* | Indicates python start port |
| *python\_end\_port* | Indicates python end port |
| *rExeLocation* | Specify absolute path for R.exe. |
| *batchCmd* | Batch command |
| *rServeLocation* | Specify absolute path for instantiateRServe.R file. This file instantiates RServe at the given port. |
| *rSeveOutputLocation* | Location for output of instantiateRServe.R. |
| *r\_start\_port* | Indicates R start port.(=) |
| *r\_end\_port* | Indicates R end port. (<=) |
| *python\_always\_open* | Indicates number of PServe always open |
| *r\_always\_open* | Indicates number of RServe always open |
|  |  |

# Consuming MLEngine

To use MlEngine in any data science project, following steps to be followed.

Prerequisite

* R package to be executed via MlEngine , should be installed as the package (on the local machine/dev/prod /server).
* Python code which is to be executed, should be placed in the same location as the python sockServe.py i.e. at location specified by *pythonSockServeLocation property in config file*

## MlEngine Client using API

Steps to be followed are:

1. Include MLEngineClient.jar in your application. It contains request and response objects for MLEngine.
2. Create a web client passing appropriate url for MlEngine webservice and jacksonJaxbJsonProvider as a parameter.
3. The url will typically be :
   1. For RService : http://<hostname:port>/MlEngine/services/RService.
   2. For PyService : http://<hostname:port>/MlEngine/services/PythonService
   3. When MlEngine is running on your local machine : hostname will be localhost and port on which tomcat is running.
   4. In case it is dev server : hostname will be <http://w00526n0v:8989//MLEngine/services/XXX>
4. Populate Request object with the required parameters.
   1. Python
      1. PyRequest object needs to be populated.
      2. Contains list of commands which are to be executed.
   2. R
      1. RRequest object needs to be populated.
      2. It should be populated with commands to be executed and input .
5. Specify the format accepted and passed by the client.
6. Invoke post method on the client passing the request object. Fetch the response method
7. Extract response object from the output fetched in the above step using readEntity.

Kindly refer MlEngineInvokerSample project sample R/Python client.

# Using POSTMAN

Service can be used/ tested using POSTMAN chrome extension as well.

# Enhancements /Changes

* Convert it to maven.

# TODO

* NEXEN API on NEXEN Lab
* Code review: with fresh perspective
* Testing Report – with analysis

hostname=localhost

pythonSockServeLocation=C:\\reference\\ModifiedSockServer.py

python\_start\_port=8100

python\_end\_port=8400

python\_always\_open=5

rExeLocation=D:\\Software\\R-3.1.2\\bin\\R.exe

batchCmd=CMD BATCH --no-save --no-restore

rServeLocation=C:\\reference\\InstantiateRServe.R

rSeveOutputLocation=C:\\reference\\

r\_start\_port=8500

r\_end\_port=8999

r\_always\_open=10

python\_max\_connection=100