In [1]:

```
from SPARQLWrapper import SPARQLWrapper, JSON , XML
```

In [2]:

```
from eventregistry import *
import json, os, sys
```

In [3]:

```
#er = EventRegistry(allowUseOfArchive=False)
er = EventRegistry(apiKey = "APIKEY-HERE")
```

```
using user provided API key for making requests
Event Registry host: http://eventregistry.org
Text analytics host: http://analytics.eventregistry.org
```

In [4]:

```
sparql = SPARQLWrapper("http://eventkginterface.l3s.uni-hannover.de/sparql")
print(sparql)
```

```
<SPARQLWrapper.Wrapper.SPARQLWrapper object at 0x000002080E7C6080>
{"_defaultGraph" : None,
"_defaultReturnFormat" : 'xml',
"agent" : 'sparqlwrapper 1.8.4 (rdflib.github.io/sparqlwrapper)',
"customHttpHeaders" : {},
"endpoint" : 'http://eventkginterface.l3s.uni-hannover.de/sparql',
"http_auth" : 'BASIC',
"method" : 'GET',
"onlyConneg" : False,
"parameters" : {},
"passwd" : None,
"queryString" : 'SELECT * WHERE{ ?s ?p ?o }',
"queryType" : 'SELECT',
"requestMethod" : 'urlencoded',
"returnFormat" : 'xml',
"timeout" : None,
"updateEndpoint" : 'http://eventkginterface.l3s.uni-hannover.de/sparql',
"user" : None}
```

In [19]:

```
sparql_query = input("Please enter the sparql query :")
```

```
Please enter the sparql query :PREFIX eventKG-s: <http://eventKG.l3s.uni-hannover.de/schema/> PREFIX ev
entKG-g: <http://eventKG.l3s.uni-hannover.de/graph/> PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-synt
ax-ns#> PREFIX so: <http://schema.org/> PREFIX sem: <http://semanticweb.cs.vu.nl/2009/11/sem/> PREFIX o
wl: <http://www.w3.org/2002/07/owl#> PREFIX dbr: <http://dbpedia.org/resource/>  SELECT ?location WHERE
{  ?event rdf:type sem:Event .  ?event owl:sameAs dbr:Battle_of_Trafalgar .  GRAPH eventKG-g:event_kg {
?event sem:hasPlace ?loc} .  GRAPH eventKG-g:dbpedia_en { ?loc owl:sameAs ?location . } }
```

In [ ]:

```
"""
def integrated_model(sparql_query):

    event_kg_result = event_kg(sparql_query)
    #print( "Event KG response : ",event_kg_result )
    event_preprocess(sparql_query)
    event_registry_to_kg(event)
    event_registry_key_extraction(event_output)
    event_registry_to_event_kg_mapper()

""""
```

In [20]:

```
def event_kg(sparql_query):
    sparql.setQuery(sparql_query)
    sparql.setReturnFormat(XML)
    results = sparql.query().convert()
    print(results.toxml())
    return results.toxml()
```

In [21]:

```
event_kg_output = event_kg(sparql_query)
print("The ouput from EventKg is as follows : ")
print(event_kg_output)
```

```
<?xml version="1.0" ?><sparql xmlns="http://www.w3.org/2005/sparql-results#" xmlns:xsi="http://www.w3.o
rg/2001/XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/2001/sw/DataAccess/rf1/result2.xsd">
 <head>
  <variable name="location"/>
 </head>
 <results distinct="false" ordered="true">
  <result>
   <binding name="location"><uri>http://dbpedia.org/resource/Atlantic_Ocean</uri></binding>
  </result>
  <result>
   <binding name="location"><uri>http://dbpedia.org/resource/First_French_Empire</uri></binding>
  </result>
  <result>
   <binding name="location"><uri>http://dbpedia.org/resource/Strait_of_Gibraltar</uri></binding>
  </result>
  <result>
   <binding name="location"><uri>http://dbpedia.org/resource/Cape_Trafalgar</uri></binding>
  </result>
  <result>
   <binding name="location"><uri>http://dbpedia.org/resource/Gulf_of_Cádiz</uri></binding>
  </result>
 </results>
</sparql>
```

```
The ouput from EventKg is as follows :
```

```
<?xml version="1.0" ?><sparql xmlns="http://www.w3.org/2005/sparql-results#" xmlns:xsi="http://www.w3.o
rg/2001/XMLSchema-instance" xsi:schemaLocation="http://www.w3.org/2001/sw/DataAccess/rf1/result2.xsd">
 <head>
  <variable name="location"/>
 </head>
 <results distinct="false" ordered="true">
  <result>
   <binding name="location"><uri>http://dbpedia.org/resource/Atlantic_Ocean</uri></binding>
  </result>
  <result>
   <binding name="location"><uri>http://dbpedia.org/resource/First_French_Empire</uri></binding>
  </result>
  <result>
   <binding name="location"><uri>http://dbpedia.org/resource/Strait_of_Gibraltar</uri></binding>
  </result>
  <result>
   <binding name="location"><uri>http://dbpedia.org/resource/Cape_Trafalgar</uri></binding>
  </result>
  <result>
   <binding name="location"><uri>http://dbpedia.org/resource/Gulf_of_Cádiz</uri></binding>
  </result>
 </results>
</sparql>
```

In [ ]:

In [22]:

```python
def event_preprocess(sparql_query):
    sparql.setQuery(sparql_query)
    #print(sparql)
    query_string = sparql.queryString

    #event = re.findall(r"sameAs dbr:(.*)",query_string)

    event = re.findall(r"dbr:([^ <]+)",query_string)
    event = "".join(event)
    event = event.replace("_", " ")
    select_string = re.findall(r"SELECT(.+?)WHERE" , query_string)
    #select_string = "".join(select_string)
    #select_string = select_string.replace("?", "")

    return event,select_string
```

In [23]:

```python
event,select_string = event_preprocess(sparql_query)
event,select_string
```

Out[23]:

```
('Battle of Trafalgar', [' ?location '])
```

In [24]:

```python
def get_select_content(select_string):
    for each_string in select_string:
        each_string = "".join(each_string)
        each_string = each_string.replace("?", "")

        return each_string
```

In [25]:

```python
select_string_processed = get_select_content(select_string)
select_string_processed = (list(select_string_processed.split(" ")))
select_string_processed
```

Out[25]:

```
['', 'location', '']
```

In [26]:

```
def event_registry_to_event_kg_mapper(select_string_processed):

    tablemap_computer = {
    'startTime' : "starttime",
    'endTime' : "endtime",
    'location' : "location"
    }
    tablemap = []
    for each in list(select_string_processed):

        if tablemap_computer.get(each) is not None:

            tablemap.append(tablemap_computer.get(each))

    return tablemap
```

In [27]:

```
entities = event_registry_to_event_kg_mapper(select_string_processed)
entities
```

Out[27]:

```
['location']
```

In [ ]:

In [28]:

```
def event_registry_to_kg(event):
    iter = QueryEventsIter(conceptUri = er.getConceptUri(event))
    for art in iter.execQuery(er, sortBy = "rel"):
        #print(art)
        event_output = (json.dumps(art, indent=4))

        break

    event_output = json.loads(event_output)

    for i in event_output["concepts"]:

        if i["type"] == "loc":
            location = (i["label"]["eng"])

            break
    starttime = event_output["eventDate"]
    endtime = event_output["eventDate"]

    return {
        "location" : location,
        "starttime" : starttime,
        "endtime" : endtime


    }
```

In [30]:

```
event_registry_output = event_registry_to_kg(event)
```

```
event_registry_output
```

Out[30]:

```
{'location': 'Civil Guard (Spain)',
 'starttime': '2018-07-22',
 'endtime': '2018-07-22'}
```

In [18]:

```
print("The ouput from event registry is as follows : ")
for ent in entities:
    if ent in event_registry_output:

        print(ent ,":" ,event_registry_output[ent])
```

```
The ouput from event registry is as follows :
location : Civil Guard (Spain)
```

In [ ]: