

Praca z HBase przy użyciu REST API i Python'a.

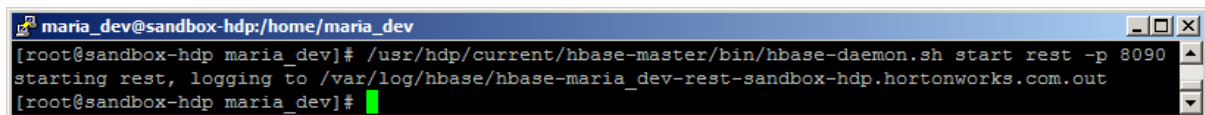
Uruchom HBase (instrukcja wyżej), jeśli jeszcze nie działa.

Zaloguj się przez Putty korzystając z loginu i hasła: **maria_dev**.

Zanim uruchomimy REST API musimy przełączyć się na superuser'a. Wpisz: **su root** i podaj hasło **danilewicz**.

Teraz możemy uruchomić REST API wpisz:

/usr/hdp/current/hbase-master/bin/hbase-daemon.sh start rest -p 8090

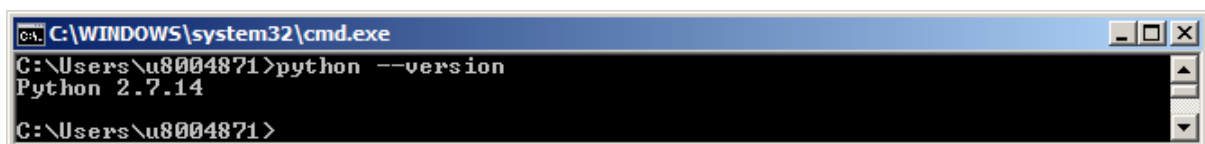


```
maria_dev@sandbox-hdp:/home/maria_dev
[root@sandbox-hdp maria_dev]# /usr/hdp/current/hbase-master/bin/hbase-daemon.sh start rest -p 8090
starting rest, logging to /var/log/hbase/hbase-maria_dev-rest-sandbox-hdp.hortonworks.com.out
[root@sandbox-hdp maria_dev]#
```

Sprawdź czy API działa wpisując w przeglądarce internetowej adres: <http://127.0.0.1:8090/>

Sprawdź czy masz zainstalowanego python'a. Otwórz wiersz poleceń w Windowsie i wpisz:

python --version



```
C:\WINDOWS\system32\cmd.exe
C:\Users\u8004871>python --version
Python 2.7.14
C:\Users\u8004871>
```

Zainstaluj teraz HBase Stargate – jest to REST API klient dla Python'a. Wpisz: **pip install starbase**

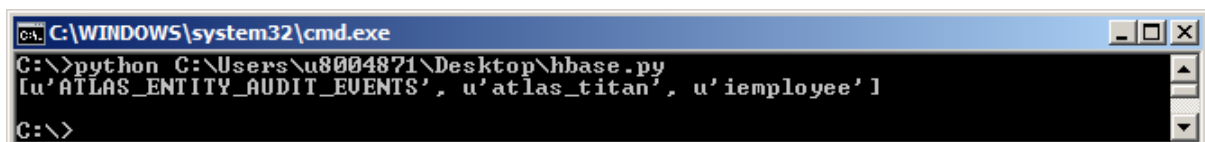
Na pulpicie utwórz plik **hbase.py**, otwórz go w notatniku i wpisz poniższy kod:

```
from starbase import Connection

conn = Connection(host='127.0.0.1', port=8090)

# pokaz tabele
tabele = conn.tables()
print(tabele)
```

Zapisz zmiany i uruchom program wpisując w wierszu poleceń python plus ścieżka do Twojego pliku na pulpicie. U mnie jest to: **python C:\Users\u8004871\Desktop\hbase.py**



```
C:\WINDOWS\system32\cmd.exe
C:\>python C:\Users\u8004871\Desktop\hbase.py
[u'ATLAS_ENTITY_AUDIT_EVENTS', u'atlas_titan', u'employee']
C:\>
```

Aby utworzyć nową tabelę dopisz tekst zaznaczony na żółto:

```
from starbase import Connection
```

```

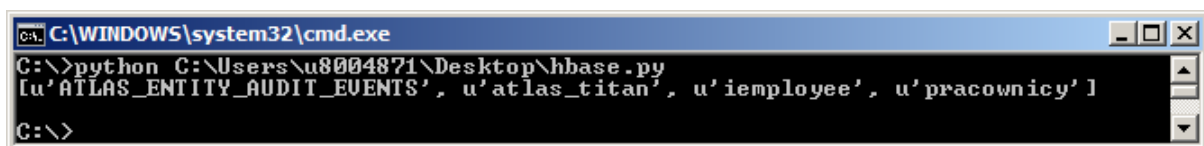
conn = Connection(host='127.0.0.1', port=8090)

# utworz instancje tabeli
tabela = conn.table('pracownicy')
# utworz tabele pracownicy z dwiema rodzinami kolumn
tabela.create('dane prywatne', 'dane sluzbowe')

# pokaz tabele
tabele = conn.tables()
print(tabele)

```

Uruchom skrypt i sprawdź czy tabela została utworzona:



```

C:\WINDOWS\system32\cmd.exe
C:\>python C:\Users\u8004871\Desktop\hbase.py
['u'ATLAS_ENTITY_AUDIT_EVENTS', u'atlas_titan', u'iemployee', u'pracownicy']
C:\>

```

Teraz zanim utworzymy tabele pracownicy, sprawdzimy czy ona istnieje i jeśli tak to ją usuniemy:

```

conn = Connection(host='127.0.0.1', port=8090)

# utworz instancje tabeli
tabela = conn.table('pracownicy')

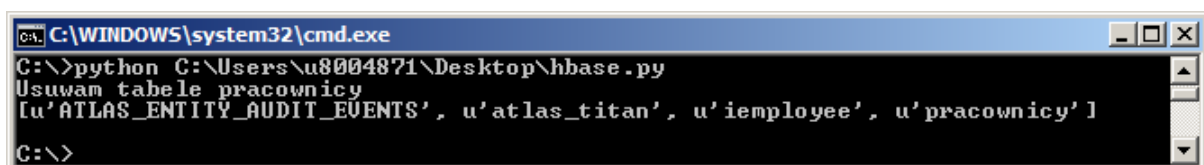
# sprawdz czy tabela istnieje
if tabela.exists():
    print('Usuam tabele pracownicy')
    tabela.drop()

# utworz tabele pracownicy z dwiema rodzinami kolumn
tabela.create('dane prywatne', 'dane sluzbowe')

# pokaz tabele
tabele = conn.tables()
print(tabele)

```

Uruchom skrypt i sprawdź czy tabela jest kasowana przed utworzeniem jeśli istnieje:



```

C:\WINDOWS\system32\cmd.exe
C:\>python C:\Users\u8004871\Desktop\hbase.py
Usuam tabele pracownicy
['u'ATLAS_ENTITY_AUDIT_EVENTS', u'atlas_titan', u'iemployee', u'pracownicy']
C:\>

```

Teraz po utworzeniu tabeli dodamy i usuniemy rodzinę kolumn o nazwie **dane tajne**:

```

conn = Connection(host='127.0.0.1', port=8090)

# utworz instancje tabeli
tabela = conn.table('pracownicy')

```

```

# sprawdź czy tabela istnieje
if tabela.exists():
    print('Usuwanie tabeli pracownicy')
    tabela.drop()

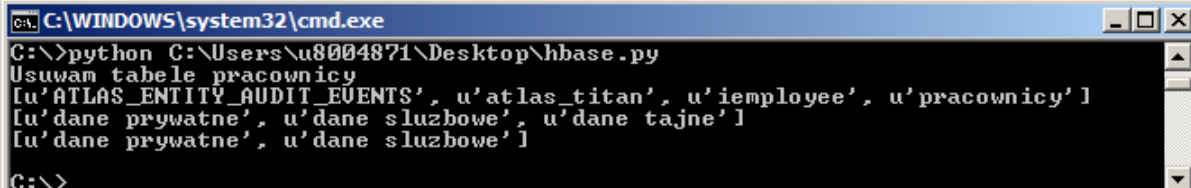
# utwórz tabelę pracownicy z dwiema rodzinami kolumn
tabela.create('dane prywatne', 'dane służbowe')

# pokaż tabelę
tabele = conn.tables()
print(tabele)

# dodaj rodzinę kolumn dane tajne
tabela.add_columns('dane tajne')
# wyświetl kolumny w tabeli
print(tabela.columns())
# usuń rodzinę kolumn dane tajne
tabela.drop_columns('dane tajne')
# wyświetl kolumny w tabeli
print(tabela.columns())

```

Uruchom skrypt i sprawdź wynik:



```

C:\WINDOWS\system32\cmd.exe
C:\>python C:\Users\u8004871\Desktop\hbase.py
Usuwanie tabeli pracownicy
[u'ATLAS_ENTITY_AUDIT_EVENTS', u'atlas_titan', u'iemployee', u'pracownicy']
[u'dane prywatne', u'dane służbowe', u'dane tajne']
[u'dane prywatne', u'dane służbowe']
C:\>

```

Teraz dodamy wiersz do naszej tabeli, a następnie go wyświetlimy :

```

...
# usuń rodzinę kolumn dane tajne
tabela.drop_columns('dane tajne')
# wyświetl kolumny w tabeli
print(tabela.columns())

# dodaj pierwszy wiersz do tabeli
tabela.insert(
    '1',
    {
        'dane prywatne': {'imie': 'Jan', 'nazwisko': 'Nowak', 'miasto': 'warszawa'},
        'dane służbowe': {'stanowisko': 'programista', 'zarobki': '200000'}
    }
)

# wyświetl wiersz o id = 1
print(tabela.fetch('1'))

```

Oto wynik:

```
C:\WINDOWS\system32\cmd.exe
C:\>python C:\Users\u8004871\Desktop\hbase.py
Usuwanie tabeli pracownicy
[('ATLAS_ENTITY_AUDIT_EVENTS', 'atlas_titan', 'employee', 'pracownicy')]
[('dane prywatne', 'dane sluzbowe', 'dane tajne')]
[('dane prywatne', 'dane sluzbowe')]
{'dane prywatne': {'imie': 'Jan', 'nazwisko': 'Nowak', 'miasto': 'warszawa'}, 'dane sluzbowe': {'stanowisko': 'programista', 'zarobki': '200000'}}
C:\>
```

W następnym kroku zaktualizujemy miasto na Warszawa i wyświetlimy tylko zaktualizowaną rodzinę kolumn i kolumnę:

```
...

# wyświetl wiersz o id = 1
print(tabela.fetch('1'))

# zaktualizuj miasto w wierszu o id = 1
tabela.update(
    '1',
    {'dane prywatne': {'miasto': 'Warszawa'}}
)

# wyświetl rodzinę kolumn 'dane prywatne' w wierszu o id = 1
print(tabela.fetch('1', ['dane prywatne']))
# wyświetl kolumnę 'miasto' z rodziny kolumn 'dane prywatne' z wiersza o id = 1
print(tabela.fetch('1', {'dane prywatne': ['miasto']}))
```

Wynik naszego programu wygląda teraz następująco:

```
C:\WINDOWS\system32\cmd.exe
C:\>python C:\Users\u8004871\Desktop\hbase.py
Usuwanie tabeli pracownicy
[('ATLAS_ENTITY_AUDIT_EVENTS', 'atlas_titan', 'employee', 'pracownicy')]
[('dane prywatne', 'dane sluzbowe', 'dane tajne')]
[('dane prywatne', 'dane sluzbowe')]
{'dane prywatne': {'imie': 'Jan', 'nazwisko': 'Nowak', 'miasto': 'warszawa'}, 'dane sluzbowe': {'stanowisko': 'programista', 'zarobki': '200000'}}
{'dane prywatne': {'imie': 'Jan', 'nazwisko': 'Nowak', 'miasto': 'Warszawa'}}
{'dane prywatne': {'miasto': 'Warszawa'}}
C:\>
```

Teraz przećwiczymy usuwanie kolumny, rodziny kolumn i całego wiersza:

```
# wyświetl rodzinę kolumn 'dane prywatne' w wierszu o id = 1
print(tabela.fetch('1', ['dane prywatne']))
# wyświetl kolumnę 'miasto' z rodziny kolumn 'dane prywatne' z wiersza o id = 1
print(tabela.fetch('1', {'dane prywatne': ['miasto']}))

# usuwamy kolumnę 'miasto'
tabela.remove('1', 'dane prywatne', 'miasto')
print(tabela.fetch('1'))

# usuwamy rodzinę kolumn 'dane prywatne'
tabela.remove('1', 'dane prywatne')
```

```
# usuwamy caly wiersz o id =1
tabela.remove('1')
print(tabela. fetch('1'))
```

```
C:\WINDOWS\system32\cmd.exe
C:\>python C:\Users\q8004871\Desktop\hbase.py
Usuwam table pracomowcy
lu ATLAS_ENTITY_AUDIT_EVENTS', u'atlas_titan', u'employee', u'pracownicy']
u'dane prywatne': u'dane sluzbowe', u'dane tajne']
lu dane prywatne: u'dane sluzbowe']
<dictproxy {'<imie': 'Jan', 'nazwisko': 'Nowak', 'miasto': 'Warszawa', 'dane sluzbowe': {'stanowisko': 'programista', 'zarobki': '200000'}}
<dictproxy {'<imie': 'Jan', 'nazwisko': 'Nowak', 'miasto': 'Warszawa'}}
<dictproxy {'<miasto': 'Warszawa'}}
<dictproxy {'<imie': 'Jan', 'nazwisko': 'Nowak', 'dane sluzbowe': {'stanowisko': 'programista', 'zarobki': '200000'}}
<dictproxy {'<dane sluzbowe': {'stanowisko': 'programista', 'zarobki': '200000'}}
None
C:\>
```

```
# usuwamy cały wiersz o id =1
tabela.remove('1')
print(tabela.fetch('1'))

b = tabela.batch()

for i in range(0, 5):
    b.insert(str(i), {'dane prywatne': {'imie': 'Adam'}})

b.commit(finalize=True)
print(tabela.fetch('4'))
```

```
C:\WINDOWS\system32\cmd.exe
C:\>python C:\Users\m8004871\Desktop\jhbase.py
C:\>python C:\Users\m8004871\Desktop\jhbase.py
[{'dane prywatne': 'u'dane sluzbowe', 'u'atlas_titan', 'u'employee', 'u'pracownicy'}
[{'dane prywatne': 'u'dane sluzbowe', 'u'dane tajne'}
[{'dane prywatne': 'u'dane sluzbowe'}
[{'dane prywatne': {'imie': 'Jan', 'nazwisko': 'Nowak', 'miasto': 'Warszawa'}, 'dane sluzbowe': {'stanowisko': 'programista', 'zarobki': '200000'}}
[{'dane prywatne': {'imie': 'Jan', 'nazwisko': 'Nowak', 'miasto': 'Warszawa'}}
[{'dane prywatne': {'miasto': 'Warszawa'}}
[{'dane prywatne': {'imie': 'Jan', 'nazwisko': 'Nowak', 'dane sluzbowe': {'stanowisko': 'programista', 'zarobki': '200000'}}
[{'dane sluzbowe': {'stanowisko': 'programista', 'zarobki': '200000'}}
None
[{'dane prywatne': {'imie': 'Adam'}}
C:\>
```