**Mule Caching Strategy With REDIS Cache server**

The objective of this tutorial is to demonstrate the implementation of Mule caching strategy with REDIS cache server.

**REDIS Server:**

1. REDIS stands for Remote Dictionary Server.
2. Stores the data in key-value pair format.
3. Can be used as Cache and message broker.
4. Server can be download from https://github.com/MicrosoftArchive/redis/releases

First we need start redis-server.exe through administrator permissions. Next we need to start redis-cli.exe with administrator permissions. Redis Server listens at port number 6739 in our localhost.

**Basic Commands:**

1. set key value [EX seconds] [PX milliseconds] [NX|XX] (To set the key and value including time in seconds)

EX: set company eidiko 1500

2. get key (To get the value based on key )

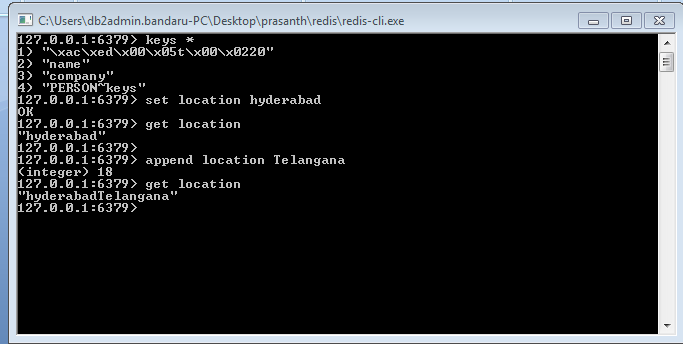
Ex: get company

3.keys pattern(to search keys based on pattern)

EX: keys \*

4.append key value(to append value to the key)

**Below is the example screenshot for Basic Commands:**

****

**Cache Strategy Configuration:**

We need to define java class by implementing org.mule.api.store.ObjectStore and overriding all the methods.The below is the java class which defines customized cache strategy.

**import** java.io.Serializable;

**import** java.util.List;

**import** org.mule.DefaultMuleEvent;

**import** org.mule.DefaultMuleMessage;

**import** org.mule.api.MuleContext;

**import** org.mule.api.MuleEvent;

**import** org.mule.api.construct.FlowConstruct;

**import** org.mule.api.context.MuleContextAware;

**import** org.mule.api.registry.RegistrationException;

**import** org.mule.api.store.ObjectStore;

**import** org.mule.api.store.ObjectStoreException;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.data.redis.cache.RedisCache;

**import** org.springframework.data.redis.cache.RedisCacheManager;

**public** **class** RedisObjectStore<T **extends** Serializable> **implements** ObjectStore<T>, MuleContextAware {

@Autowired

**private** RedisCacheManager cacheManager;

**private** org.springframework.data.redis.cache.RedisCache cache;

**private** MuleContext context;

**private** DefaultMuleMessage message;

@Autowired

**public** **void** setCache() {

**this**.cache = (RedisCache) **this**.cacheManager.getCache("PERSON");

}

**public** RedisCache getCache() {

**return** **this**.cache;

}

@Override

**public** **synchronized** **boolean** contains(Serializable key) **throws** ObjectStoreException {

System.***out***.println("Inside contains Method");

**if** (cache.get(key.toString(), Object.**class**) == **null**)

**return** **false**;

**else**

**return** **true**;

}

@Override

**public** **synchronized** **void** store(Serializable key, T value) **throws** ObjectStoreException {

System.***out***.println("Inside Store Method");

MuleEvent event = (MuleEvent) value;

@SuppressWarnings("unchecked")

List<Object> person = (List<Object>) event.getMessage().getPayload();

cache.put(key.toString(), person);

}

@SuppressWarnings("unchecked")

@Override

**public** **synchronized** T retrieve(Serializable key) **throws** ObjectStoreException {

System.***out***.println("Inside Retrieve Method");

List<Object> person = (List<Object>) cache.get(key.toString(), Object.**class**);

DefaultMuleEvent event = **null**;

String operation = **null**;

operation = context.getRegistry().get("operation");

**if** (operation.equalsIgnoreCase("store")) {

**return** **null**;

} **else** **if** (person == **null** && operation.equalsIgnoreCase("remove")) {

**return** **null**;

} **else** **if** (person == **null** && operation.equalsIgnoreCase("retrieve")) {

message = **new** DefaultMuleMessage("Key " + key.toString() + " not found", context);

FlowConstruct flow = context.getRegistry().lookupFlowConstruct("cacheFlow");

event = **new** DefaultMuleEvent(message, org.mule.MessageExchangePattern.***ONE\_WAY***, flow);

} **else** {

message = **new** DefaultMuleMessage(person, context);

FlowConstruct flow = context.getRegistry().lookupFlowConstruct("cacheFlow");

event = **new** DefaultMuleEvent(message, org.mule.MessageExchangePattern.***ONE\_WAY***, flow);

}

**return** (T) event;

}

**public** **synchronized** T remove(Serializable key) **throws** ObjectStoreException {

T value = retrieve(key);

**if** (value != **null**) {

cache.evict(key);

**try** {

context.getRegistry().registerObject("evict", "Key " + key.toString() + " evicted from cache");

} **catch** (RegistrationException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

} **else** {

**try** {

context.getRegistry().registerObject("evict", "Key " + key.toString() + " not found");

} **catch** (RegistrationException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

**return** value;

}

@Override

**public** **boolean** isPersistent() {

**return** **true**;

}

@Override

**public** **synchronized** **void** clear() **throws** ObjectStoreException {

System.***out***.println("Inside clear method");

}

@Override

**public** **void** setMuleContext(MuleContext context) {

**this**.context = context;

}

}

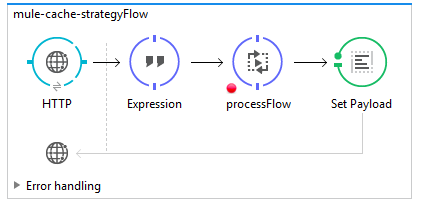
**Creating Global elements for Cache Strategy Configuration:**

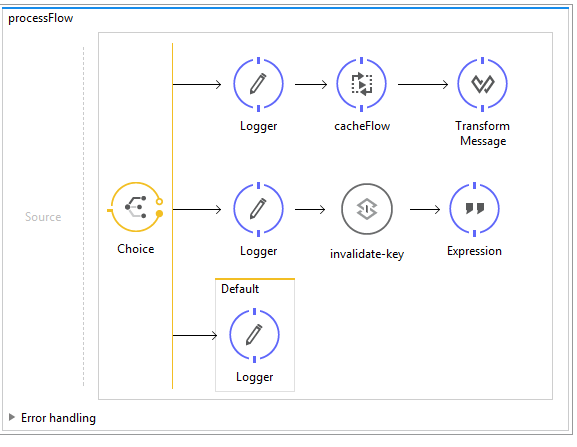
1.Select the global elements tab and click on create button and select caching strategy.

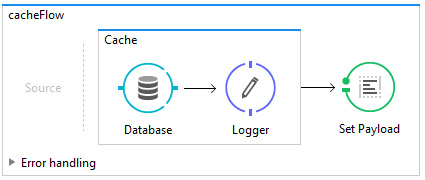
2.Define the Object store by selecting defined above class(RedisObjectStore).

3.Select Key Expression option under the Event Key section and then type the MEL #[message.inboundProperties.'http.query.params'.key] and click **OK.**

**Mule Flows:**



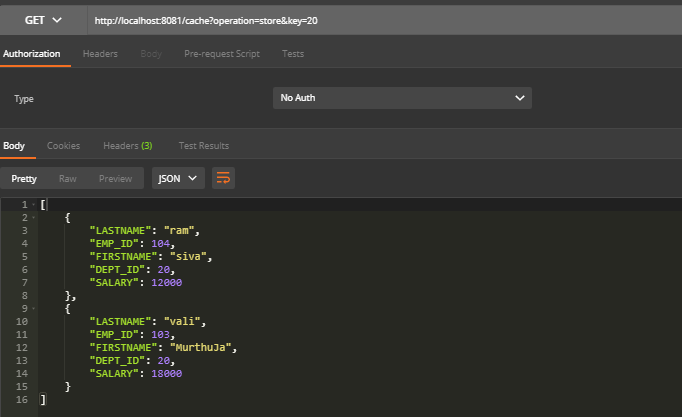


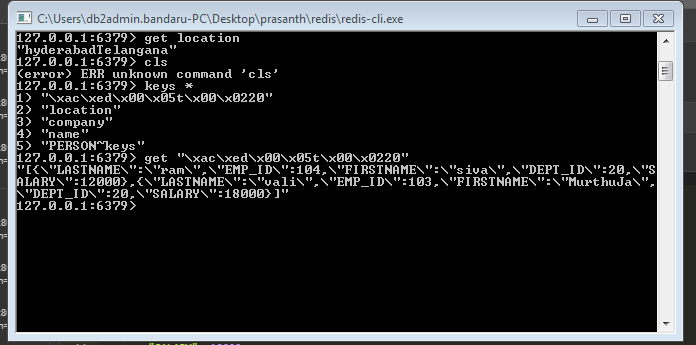


**Store Operation:**

1. http://localhost:8081?operation=store&key=20;

2.This request calls the store operation from Customized cache strategy i.e., the store method in java class and stores the records in redis cache server.

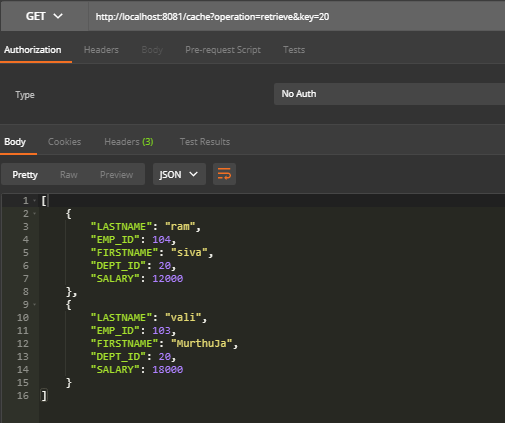




**Retrieve Operation:**

1.http://localhost:8081?operation=retrieve&key=20

2.This Request calls the retrieve operation in the java class and reads the details from cache



**Remove Operation:**

1.http://localhost:8081?operation=remove&key=20

2.This request call the remove method in java class and removes the specified details from cache

