

Bringing the Pricing service up

1. Install redis & run stand along (download tar+make/homebrew/docker). 2 step instruction for docker
  - a. `docker pull redis:4.0`
  - b. `docker images (get the container ID) // docker images | grep redis`
  - c. `docker run -p 6379:6379 < CONTAINER ID e.g. bfc1f6df2db. >`
2. Star the application
  - a. `pricing-service$ java -jar pricing-service.jar`
3. [Optional-If you like to see what is happening with REDIS Queues as you execute the the api-s below, you could do `>>redis-cli monitor`

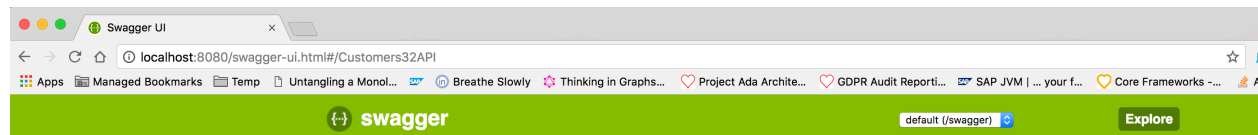
Endpoints & Swagger UI:

1. **Swagger UI in the App:**
  - a. Use Swagger UI or CURL for API Calls (URL: <http://localhost:8080/swagger-ui.html> (Swagger UI))
2. **Generate Seed Customer Dataset:**
  - a. POST/Customer generates 100 records per service plan for 2 countries (US/DE), totaling 600 Records. There is randomness in unique keys ID, so you can call as many times as you like.
  - b. Either curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: */*' 'http://localhost:8080/customers'
```

- c. *or Swagger UI* do POST/Customers & click tryout.

G



### Netflix Pricing Service API

API for Accessing Netflix Pricing

**Customers API** : Not the main api - used for dummy customer population and verification

		Show/Hide	List Operations	Expand Operations
GET	/customers	getAllCustomersOrByCountryCode		
POST	/customers	createDummyCustomersForS1PS		
POST	/customers/{COUNTRY_CODE}/{SERVICE_PLAN}/{PRICE}/{CURRENCY}/{COUNT}	createDummyCustomersForCountryServicePlan		
GET	/customers/{ID}	getCustomerById		

Netflix Pricing Customers API : Main REST API for accessing & rollout out Price

3. [Optional]For generating more targeted data set, use the the other Post , where you could pass the country code (2 digit country iso code, SP1S/SP2S/SP4S for service plan,

any price (old price in customer record), currency is a 3 letter code (USD/EUR), count is the number of records you want to generate

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: */*'
'http://localhost:8080/customers/IN/SP1S/299.99/INR/399'
```

4. Make API calls to update price: Keep note of country/service plan and the price you apply, so you could verify later
  - a. You can make price change for one country/one plan or to a group of plans across countries. Effective date is the time at which price must be applied. Validation is disabled (Effective date can be in the past)
  - b. Curl:

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '[ \
{ \
  "countryISOCode": "US", \
  "currency": "USD", \
  "effectiveDate": "2018-05-09T23:41:24.618Z", \
  "price": 12.99, \
  "servicePlan": "SP1S" \
}, \
{ \
  "countryISOCode": "US", \
  "currency": "USD", \
  "effectiveDate": "2018-05-09T23:41:24.618Z", \
  "price": 13.99, \
  "servicePlan": "SP2S" \
}, \
{ \
  "countryISOCode": "US", \
  "currency": "USD", \
  "effectiveDate": "2018-05-09T23:41:24.618Z", \
  "price": 13.99, \
  "servicePlan": "SP2S" \
}, \
{ \
  "countryISOCode": "DE", \
  "currency": "EUR", \
  "effectiveDate": "2018-05-09T23:41:24.618Z", \
  "price": 11.98, \
  "servicePlan": "SP1S" \
} \
\
] \
' 'http://localhost:8080/prices'
```

## a. Swagger UI

Customers API : Not the main api - used for getting customer population and verification

Netflix Pricing Customers API : Main REST API for accessing & rollout out Price		
	Show/Hide	List Operations Expand Operations
DELETE	/prices	deletePriceChangeEntryIfNotActive
GET	/prices	getAllServicePlanPricesForAllCountries
POST	/prices	updatePrices
GET	/prices/{COUNTRY}	getServicePlanPriceByCountry
GET	/prices/{COUNTRY}/{PLAN}	getServicePlanPriceByCountryAndServicePlan
[ BASE URL: / ]		

## b. Price change request body

### a. One change:

```
[
  {
    "countryISOCode": "US",
    "currency": "USD",
    "effectiveDate": "2018-05-09T23:41:24.618Z",
    "price": 12.99,
    "servicePlan": "SP1S"
  },
]
```

### b. Multiple Change

```
[
  {
    "countryISOCode": "US",
    "currency": "USD",
    "effectiveDate": "2018-05-09T23:41:24.618Z",
    "price": 12.99,
    "servicePlan": "SP1S"
  },
  {
    "countryISOCode": "US",
    "currency": "USD",
    "effectiveDate": "2018-05-09T23:41:24.618Z",
    "price": 13.99,
    "servicePlan": "SP2S"
  },
  {
    "countryISOCode": "DE",
    "currency": "EUR",
    "effectiveDate": "2018-05-09T23:41:24.618Z",
    "price": 11.98,
    "servicePlan": "SP1S"
  }
]
```

- Make API calls to the Pricing server (from CURL or UI ) to verify if the updated price is marked as the active price in the Pricing Service side: `GET /prices/{COUNTRY}`. You should see the current price as active. (There could be upto 5 second delay as this flag is updated by a scheduled job).

If you apply the price for the future date, the price won't be shown as active. For the tests, consider using the current or past date/time.

```
curl -X GET --header 'Accept: application/json'
'http://localhost:8080/prices/US?active=true'
```

6. Now go the customer pricing side. Make API calls from CURL or UI to verify if the updated price is shown for the customer (GET /customers)

```
curl -X GET --header 'Accept: application/json'
'http://localhost:8080/customers?countryCode=US'
```

If you want to check for a specific customer, you could also use the GET /customers/{ID} endpoint (Before making the change get the customer id) to see the change

That's it 😊