Consider this as a micro service which supports shorten and expand functionalities.

This micro service exposes two REST apis one for shortening and another for expanding.

Let’s have a Spring RestController mapped with POST mapping for shortening the given url, since it adds data into persistence layer.

Another GET REST controller mapping for expanding the given keyword. This will be a generic mapping which looks for + sign at the end of the URL path variable.

**Suggested High Level Classes:**

It should follow typical MVC (Model View Controller pattern)

View -> not applicable here as this is exposed as REST service.

Model -> Service which speaks with database and returns expected data.

Controller -> RestController which is exposed to external world, and calls Service to do the needful.

**Service (@Service)**

ShortenService.produceNewValue(String Url)

This produces a new six characters value and stores it into database, and returns it.

ShortenService.expand(String key)

This returns the expanded (actual URL string from database).

Database has a table to store key (generated six characters), URL. Where key can be considered as a primary key.

**Caching:**

There are many ways this can be achieved.

* + 1. In memory caching
    2. Database provided caching.
    3. Custom Caching (ConcurrentHashMap)

If the database provides caching support (like couch base caching), then caching can be handled using provider’s way, if not.

Then a ConcurrentHashMap can be used to store key,URL for every insert (produceNewValue).

*Service will be updated as follows:*

ShortenService.produceNewValue(String Url)

1. Has a ConcurrentHashMap cache of Key (six characters produced key), URL String as value.

2. If URL not in the cache, then produces a new key.

3. Ensures key is not in cache if it is then produces another. If not,

then inserts key, URL into database, adds key, URL into cache. (refreshes cache)

4. Returns the key to the caller.

ShortenService.expand(String key)

Pulls the URL (value) from ConcurrentHashMap cache and returns it.

If key not available then refreshes cache by pulling latest data from database, and returns URL if available otherwise returns an exception/error or null.

**Scaling/Loan Balancing.**

Since, this has been deployed as micro service, it would be quite handy to add a Docker Image for this micro service, and deploy it into Kubernetes Cluster, and then this can be scaled up/down depending on requirement. As all instances has its own cache to refer values, and time to time the cache can be refreshed from common database.

**Suggested Server:**

Any application server can do this, but I prefer a simple Tomcat application server or Jetty server can do this.

**Server Costs:**

I have no idea.

**Assumption:**

I am not sure how this functionality is exposed, hence I presume myself as a micro service using RestControllers.

I have not covered the error part of it as I am not sure what to do with there are any errors like key not found, or using the same URL to shorten again etc.,

**As this seems to be a very simple requirement, this can also achieved using NodeJs, AngularJs etc (I am not experienced on this, I prefer Java hence my solution is given in Java and its related technologies).**

Questions:

* How exceptions are handled? if given key to expand is not valid! Or duplicate URL is given again to shorten etc.,
* I think there will be a limitation on generating six character long keys, for example 6 characters can be rearranged unto 720 times, and for this case it will be some n numbers, what will happen if generated keys crosses that limit?