-Simple product search

# The problem

We have a data about our products stored in both ElasticSearch and MySQL database. The data is same in both and we mainly query the ElasticSearch. However, from time to time we need to fetch the data directly from MySQL while we do some fancy experiments in ElasticSearch. We would like to access the data about a concrete product from our frontend so we can display it to customer.

As we do have quite some traffic we also need some kind of a caching to be implemented. A simple filesystem cache is sufficient for now but we need to be able to switch it for something more advanced in the future just by a modification of a config file.

Business always wants to know what are the most successful products. Therefore we also need to track the number of requests for each product. We are again ok with storing the information in a plain text file for now. Keep in mind that we will need to change the storage in future as well and swap it for something more robust.

# Your mission dear candidate, should you decide to accept it

You should create a new controller with a method accepting id of product as a parameter. The method should return json representation of product data.

A basic workflow for the task:

* Request with product id comes in.
* If product is cached retrieve from cache.
* If product is not cached retrieve from ElasticSearch/MySQL and add to cache.
* Increment the number of requests for given product.
* Return product data in JSON.

A controller might look like the following:

class ProductController

{

/\*\*

\* @param string $id

\* @return string

\*/

public function detail($id)

{

// do stuff and return json

}

}

# Things that might help you out

We do have drivers for both ElasticSearch and MySQL so you do not have to write them. You can assume they work and by calling their methods you get an array of data for given product and each of them implements its interface:

interface IElasticSearchDriver

{

/\*\*

\* @param string $id

\* @return array

\*/

public function findById($id);

}

interface IMySQLDriver

{

/\*\*

\* @param string $id

\* @return array

\*/

public function findProduct($id);

}

You can also safely assume that:

* The outer framework does exactly what you need it to do. If you need to pass some parameters to constructor of the controller it always passes the correct ones.
* Whatever product id is passed the drivers always find a product. You do not have to deal with "Not found" exceptions.
* The cache is infinite. You do not need to worry about the invalidation of cache. Once the data is cached we do not need remove it, ever.
* The info about number of requests for given product is just a simple pair productId => numberOfRequests, no other information is needed.