Befeni Technical Test [Advanced]

Test Overview

You are working on a project that receives shirt orders from several sources such as an external API or an in-house data warehouse. Because the import process is expensive - you have already set up a local MySQL database to store the data after the import.

Additionally you might have a cache database (e.g. Redis) for quicker access to frequently used data or an ElasticSearch instance to filter that data. The Shirt Order model in it's simplest form has a customer ID, fabric ID, collar size, chest size, waist size and wrist size. You can find a version of this model below.

Below are the requirements.

The number of hours after the requirements are an indication of the time you need to finish the task. If you need extra hours that is not a problem. The assignment is abstract on purpose.

Requirement 1.

Est. Time: 4 hours.

Maintaining the code that satisfies the above conditions is obviously going to be a difficult task, unless there is a proper design to reduce complexity and enhance flexibility.

You are requested to design an architecture where all shirt order data will be requested from a centralized point; a class called ShirtOrderRepository.

The ShirtOrderRepository is responsible to talk to the different data sources, and for that reason it needs to be able to talk to a new data source with minimal additions to the code.

It also needs to be able to select which data sources it is going to communicate with (in other words, it should

be possible to add/remove the data sources dynamically). We prefer a solution uses methods that are common whether you have mysql, cache, elasticsearch or whatever else.

Requirement 2.

Est. Time: 1 hour.

Your implementation needs to be accompanied by automated tests.

Requirement 3.

Extra, not necessary. Est. Time: 8 hours.

After the implementation of the above requirements you can now access your Shirt Order model without caring where it came from. There is one little catch though. If you decide to update/delete the model, this action needs to be reflected to all the data sources that contain this shirt order. For that reason you are requested to design the appropriate classes/code to reflect the actions that alter a model to all associated data sources.

```
<?php
namespace Befeni\Model;
/**
* A test Shirt Order model
*/
class ShirtOrder
{
    /**
     * The id of the shirt order
     *
     * @var integer
     */
    public $id;

    /**
     * The id of the customer
     *
     * @var integer
     */
    public $customerId;

    /**
     * The id of the fabric
     *
```

```
     * @var integer
     */
    public $fabricId;

    /**
     * The size of the customer's collar in inches
     *
     * @var integer
     */
    public $collarSize;

    /**
     * The size of the customer's chest in inches
     *
     * @var integer
     */
    public $chestSize;

    /**
     * The size of the customer's waist in inches
     *
     * @var integer
     */
    public $waistSize;

    /**
     * The size of the customer's wrist in inches
     *
     * @var integer
     */
    public $wristSize;

}
```