# Home Challenge

Thank you for taking the time to do our **Home Challenge.**
It has two parts.
Taking into account the experience you have, you can choose to provide the complete solution,
or to focus only on the test implementation.
**Please submit your results by sending us a link to your github repository named {firstname}-{lastname}.**

1. We would like to have a **RESTful** web service that stores some monetary transactions
   (in memory is fine, no hibernate or any other orm should be used) and returns
   information about those transactions.

2. We would like to have a testing implementation for the **RESTful API** specification below. The framework
   and the design is free of choice.
   In general we are looking for a good implementation, idea behind and solid coverage.
   Explanation of what is to be tested and what is the framework doing.
   If the service is not implemented, the execution is not taken into consideration, but only the test implementation.
   Explanation of the implementation is very welcomed.

The transactions to be stored have a type and an amount. The service should
support returning all transactions of a type. Also, transactions can be linked
to each other (using a `parent_id`) and we need to know the total amount involved for all transactions linked to a particular transaction.

In general we are looking for a good implementation, code quality and how the
implementation is tested.

# In detail the API specs look like the following:

## `PUT /transactionservice/transaction/$transact`

Body:

```
{  "amount": double, "type": string, "parent_id": long  }
```

where:

- `transaction_id` is a long specifying a new transaction
- `amount` is a double specifying the amount
- `type` is a string specifying a type of the transaction.
- `parent_id` is an optional long that may specify the parent

transaction of
this transaction.

## GET /transactionservice/transaction/$transact

Returns:

```
{  "amount": double, "type": string, "parent_id": long  }
```

## GET /transactionservice/types/$type

Returns:

```
[ long, long, .... ]
```

A JSON list of all transaction ids that share the same type `$type`.

## GET /transactionservice/sum/$transaction_id

Returns:

```
{ "sum", double }
```

A sum of all transactions that are transitively linked by their `parent_id` to `$transaction_id`.

# Some simple examples would be:

```
PUT /transactionservice/transaction/10
    { "amount": 5000, "type": "cars" }
=> { "status": "ok" }


PUT /transactionservice/transaction/11
    { "amount": 10000, "type": "shopping", "parent_id": 10 }
=> { "status": "ok" }


GET /transactionservice/types/cars
=>  [ 10 ]


GET /transactionservice/sum/10
=>  { "sum": 15000 }


GET /transactionservice/sum/11
=>  { "sum": 10000 }
```

Let us know if you have any questions.