The Flamingo Route

Assignment 4

Kai Arne S. Myklebust, Silvan Adrian

Handed in: October 10, 2018

Contents

1 Solution

1.1 Files

All Files are situated in the **src/** folder:

- **flamingo.erl** The flamingo server implmentation

- **greetings.erl** The greetings module implementation
- **hello.erl** The hello module implementation
- **mood.erl** Mood module implementation
- **counter.erl** Counter module implementation
- **test_xxx.erl** Tests for each module

1.2 Running the programm

Out of convenience we used a Emakefile which compiles all the erlang files in one go then rather compile each file on it's own. This can be done by using the erlang shell and run:

```
make:all([load]).
```

1.3 Running the tests

The tests can be run with eunit, we included tests for each module in a own file. Example running tests for flamingo:

```
eunit:test(test_flamingo, [verbose]).
```

2 Implementation

2.1 Assumptions

We assumed that empty Paths shouldn't be inputed into a routing group, therefore they get ignored. Also we decided that duplicated paths get removed from existing routing groups and get added as a new routing group. In case the path of a routing group is empty the routing group gets removed from the routing group list.

2.2 Our implementation

We chose to implement the routing groups as a list, which contains triplets consisting of List of Paths, Function and State. When a new route gets registered we traverse the routing group list and check every single path in each routing group to either update it or insert it as a new routing group.
For prefix handling we also traverse each routing group to search for the longest

matching prefix in each group, from there we take the longest out of the longest matching ones.

Sadly we weren't able to solve the issue of functions of each routing group only running consecutively.

2.3 Possible improvements

We started off doing it with a map, but ended up using a list since it was easier to handle. For big routing groups a map might be faster due to easier lookups and handling of the data. Also traversing the lists is not very nice due to lists in lists which ends up to be not easily readable.

For letting the function in a routing group not run in a concurrent fashion we would use spawn_link to keep track if a process is still running or not.

3 Assessment

3.1 Scope of Test Cases

We tested edge cases as well as happy cases, which we came up with. The only thing we weren't able to test is the error from the flamingo server when adding new routes, which we don't know how to get. Also thanks to our test cases we were able to find some bugs, for example the empty Path and empty Path list.

3.2 Correctness

We are quite happy with the result, since our test cases and the OnlineTA didn't find any more bugs or errors. So we come to the conclusion that our code is relatively stable and useable.

3.3 Code Quality

Our code is well structured and commented where needed, on the other hand our loop seems to be a little bloated which might not seem that nice. But we weren't able to find a much better solution for it, so we are quite happy with it. We followed closely the requirements specified in the assignment, which we wrote tests for. Therefore we think that the functionality matches the requirements.

A Code Listing

B Tests Listing