Assignment:

---

To test your skills in DOM parsing and REST principles, we want you to write a live scraper for Amazon Prime.

By live scraper, we specifically mean a service that will make a background request, fetching the respective Amazon web site, parsing it and giving back a valid result to the requester - in this way building your own small API to Amazon Prime movies. When requesting an Amazon ID from this API, we want to get back accessible and meaningful results in the JSON format provided below.

Please do not use a database to store or query intermediate results. The result should be fetched and calculated in real time (several seconds is okay, of course).

The scraper should act as a REST API, listening on http://localhost:8080.

Example URLs to be fetched:

http://www.amazon.de/gp/product/B00KY1U7GM

http://www.amazon.de/gp/product/B00K19SD8Q

The last part of the URL is called an Amazon ID: `B00KY1U7GM`

When requesting an Amazon ID via this API at the route "http://localhost:8080/movie/amazon/{amazon_id}", the API needs to return the following JSON with the correct MIME type, when using the amazon_id `B00K19SD8Q`:

{

"title": "Um Jeden Preis",

"release_year": 2013,

"actors": ["Dennis Quaid","Zac Efron"],

"poster": "http://ecx.images-amazon.com/images/I/51UZ8st2OdL._SX200_QL80_.jpg",

"similar_ids": ["B00SWDQPOC","B00RBPBO1G","B00S2EMECI","B00M5GH53M","B00IH8BA3S", "B00M5JP1DA"]

}

Note that title language or similar movie IDs may change based on time

and location, so the above JSON is just an example.

Use the Go language and any packages you like, we will execute a `go get ./... -v` before building your application. Your live scraper should not depend on any third party services or databases other than a working internet connection to the Amazon web site.

Please upload your scraper as a *private* Git repository to github.com and give access to `falschparker82` and `zwopir` before notifying us about your entry.

The criteria on which we will assess your entry include all of the following in descending order of importance:

- Code working to specification

- Time taken for the assignment

- Simplicity and clarity

- Code Style including

- naming

- file and package structure

- (appropriate) use of Go specific features

- comments and comment quality

- Use of standard library

- Selection and use of 3rd party libraries

- Repository style and use of standards

- Performance

- Resource use and allocation patterns