Webengineering

2017-05-08

Dr. Michael Lesniak

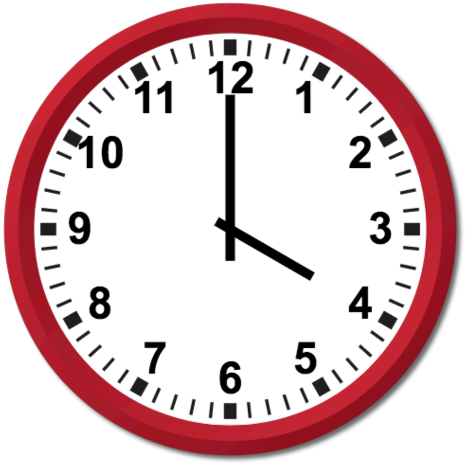
mlesniak@micromata.de

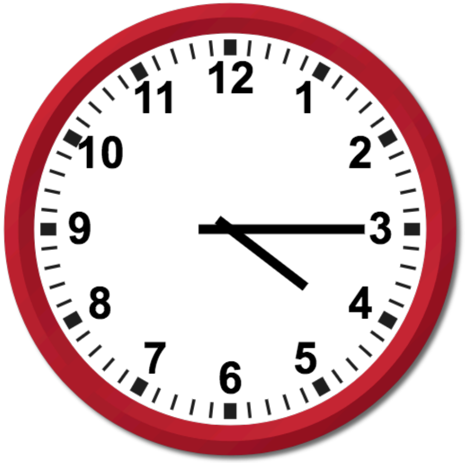
German or english – part two

- Observation
  - 80% german and english
  - 10% german > english
  - 10% englisch > german

- My current preference
  - slides in english
  - lecture in german
  - idea: more class participation?
  - documentation for everything online is still in english
  - you decide…

Lecture starts at...?





Roadmap 2017-04-24

- 16:00 c.t. or 16:00 s.t.
- German or english?
- Response to feedback
- Best practice: git commit messages
- Solution to exercises

- Persistence

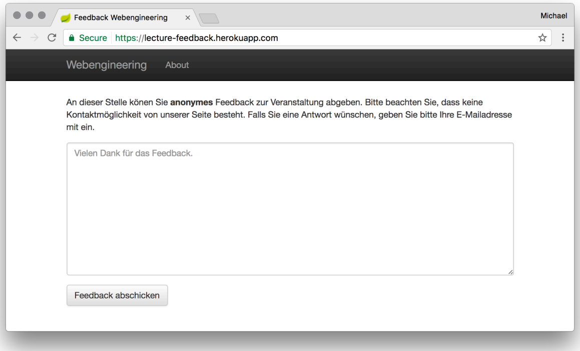
- Exercises

Response to feedback

- No feedback given
- Great. Everything is perfect! Or is it…?

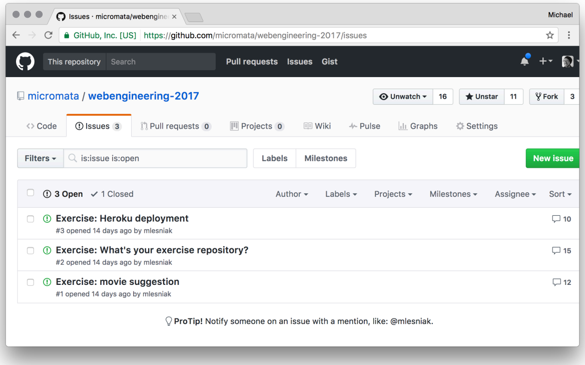
- Use the options I gave you
  - ask questions (in lecture, online, …)
  - give feedback
  - shape this lecture

- Counting…
  - 11 stars
  - 16 repositories on [github]()
  - students.size() <= 16 ?



An dieser Stelle könen Sie **anonymes** Feedback zur Veranstaltung abgeben. Bitte beachten Sie, dass keine Kontaktmöglichkeit von unserer Seite besteht. Falls Sie eine Antwort wünschen, geben Sie bitte Ihre E-Mailadresse mit ein.

Vielen Dank für das Feedback.



Exercise: Heroku deployment — 10
#3 opened 14 days ago by mlesniak

Exercise: What's your exercise repository? — 15
#2 opened 14 days ago by mlesniak

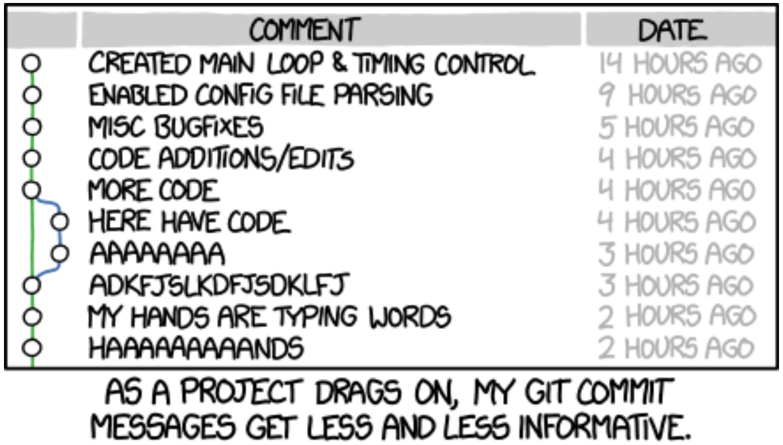
Exercise: movie suggestion — 12
#1 opened 14 days ago by mlesniak

ProTip! Notify someone on an issue with a mention, like: @mlesniak.

Git Commit Messages >> Motivation

As all best practices, it‘s just that – not a global rule!

- a well-crafted Git commit message
  - communicates context about a change
  - to colleagues
  - to future self
- A diff will tell you **what** changed
- A commit message will tell you **why**



	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Source https://chris.beams.io/posts/git-commit/

Git Commit Messages >> 7 rules

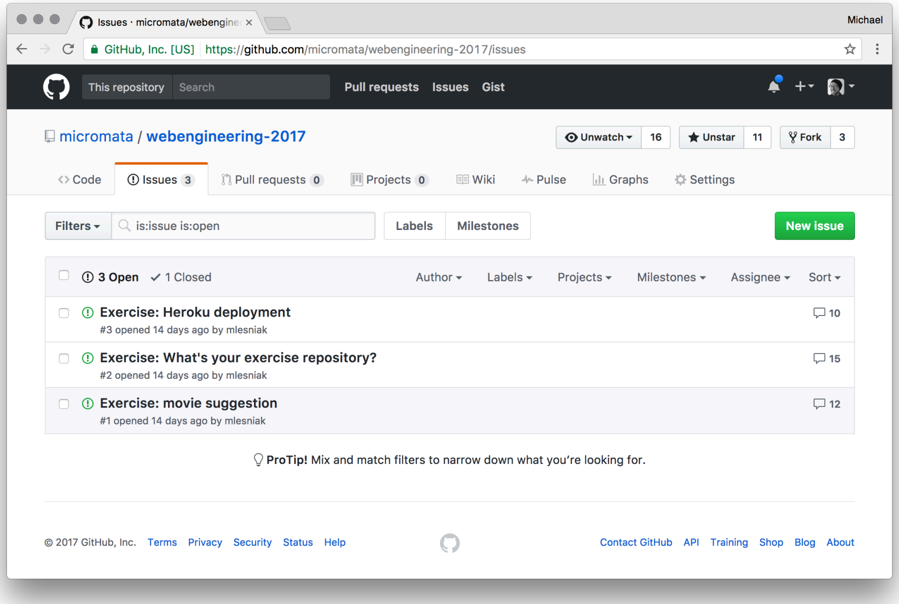
1. Separate subject from body with a blank line
2. Limit the subject line to 50 characters
3. Capitalize the subject line
4. Do not end the subject line with a period
5. Use the imperative mood in the subject line

   „This commit will <your subject line here>…“

6. Wrap the body at 72 characters
7. Use the body to explain what and why vs. how

Solution >> GitHub

- GitHub
  - Create an account (or use your existing one)
  - Write your movie suggestion
  - Star the repository



Exercise: Heroku deployment
#3 opened 14 days ago by mlesniak — 10

Exercise: What's your exercise repository?
#2 opened 14 days ago by mlesniak — 15

Exercise: movie suggestion
#1 opened 14 days ago by mlesniak — 12

ProTip! Mix and match filters to narrow down what you're looking for.

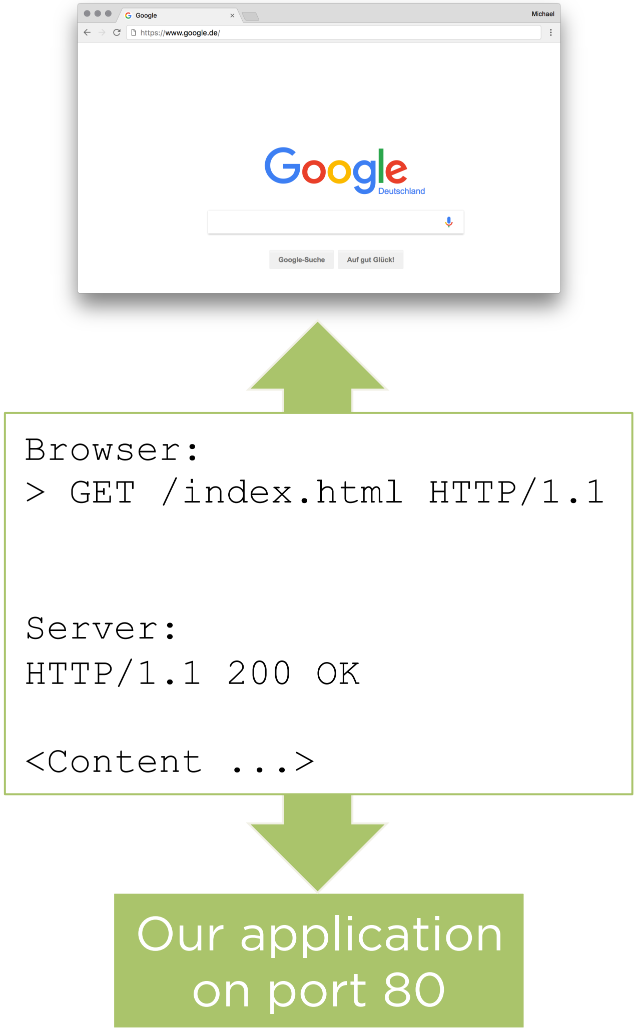
Solution >> Premark

- TIMTOWTDI
- Let‘s discuss (...albeit I might move the discussion to the exercise)
- Think about your solution
  - What is different?
  - What was your line of thinking?
  - What is better with your solution? What is better with mine?
  - Remember: every solution has trade-offs!
- Feedback request
  - Difficulty? The better I know the difficulty the better I can estimate the difficulty of the lecture project...
  - Do you want the solutions explained or are my commits sufficient?

Solution >> What is HTTP?

- HTTP
  - Hypertext Transfer Protocol
  - is the technical protocol to exchange data between web browsers and servers
  - usually on port 80

- Message
  - Header
  - Body



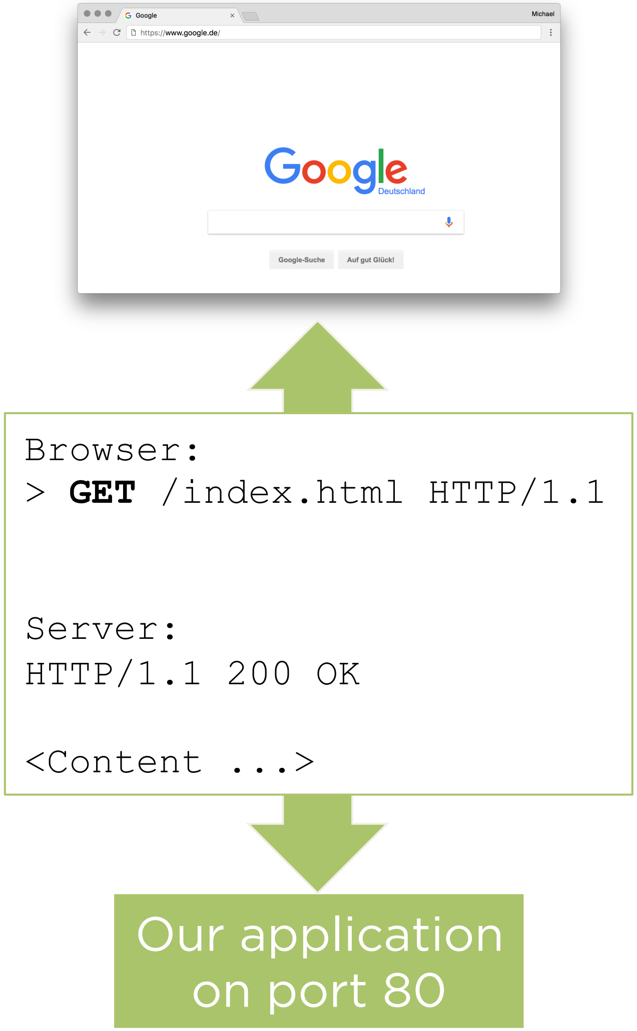
```
Browser:
> GET /index.html HTTP/1.1


Server:
HTTP/1.1 200 OK

<Content ...>
```

Our application on port 80

Sources [HTTP]

Solution >> HTTP verbs

- GET
- HEAD
- POST
- PUT
- DELETE
- TRACE
- OPTIONS
- CONNECT
- PATCH



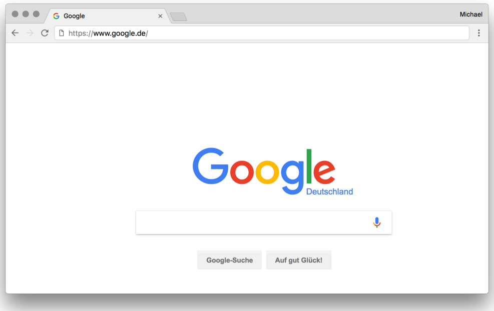
```
Browser:
> GET /index.html HTTP/1.1


Server:
HTTP/1.1 200 OK

<Content ...>
```

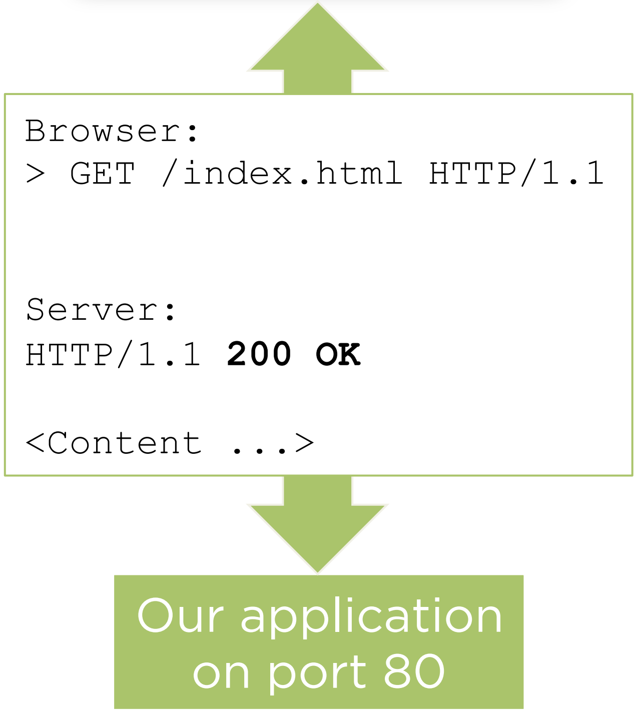
Our application on port 80

Sources HTTP

Solution >> HTTP status

- Status information for the response
- Divided into groups
  - 1xx information
  - 2xx Success
    - 200 OK
    - 201 CREATED
  - 3xx Redirection
  - 4xx Client error
    - 401 UNAUTHORIZED
    - 418 … look it up
  - 5xx Server error
    - 500 INTERNAL SERVER ERROR





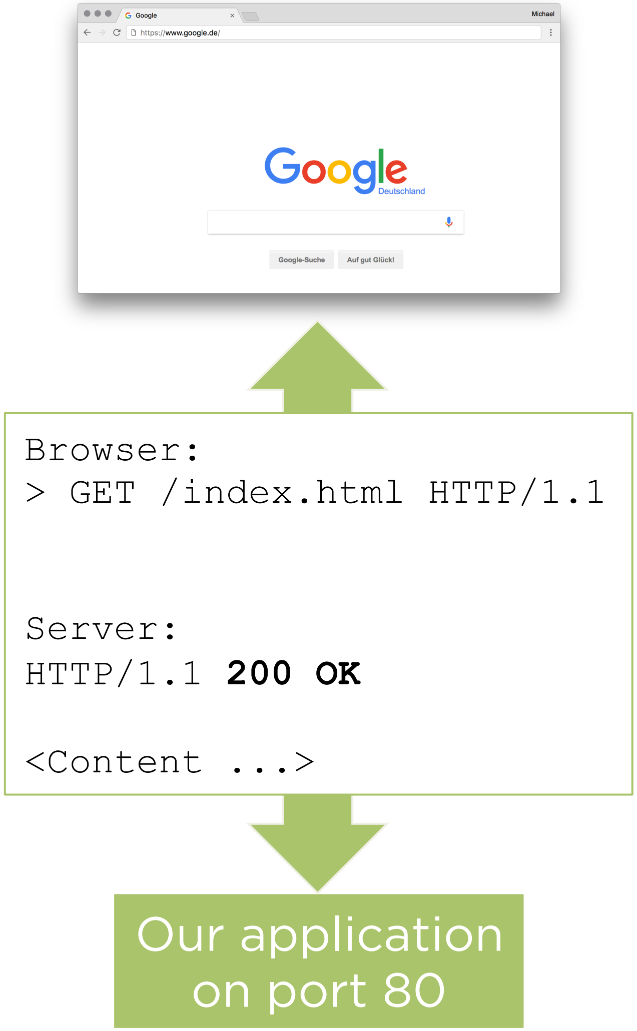
```
Browser:
> GET /index.html HTTP/1.1


Server:
HTTP/1.1 200 OK

<Content ...>
```

Our application on port 80

Sources HTTP

Solution >> HTTP: How to see what‘s happening

- Command line tool httpie
- {Chrome, Firefox, Safari} Dev Tools



```
Browser:
> GET /index.html HTTP/1.1


Server:
HTTP/1.1 200 OK

<Content ...>
```

Our application on port 80

Solution >> Improve post data structure

- Use a POJO instead of a simple String to represent posts.
- Add time of creation to each post
- Check that the time is returned in the post list
- => Code

Solution >> Improve adding new posts

- Is there a better HTTP verb (approach) for adding new posts?
- HTTP is about resources (see also REST)
- In our case: posts
- Create a new resource (entity) with HTTP POST
- => Code

Solution >> Retrieve a single post

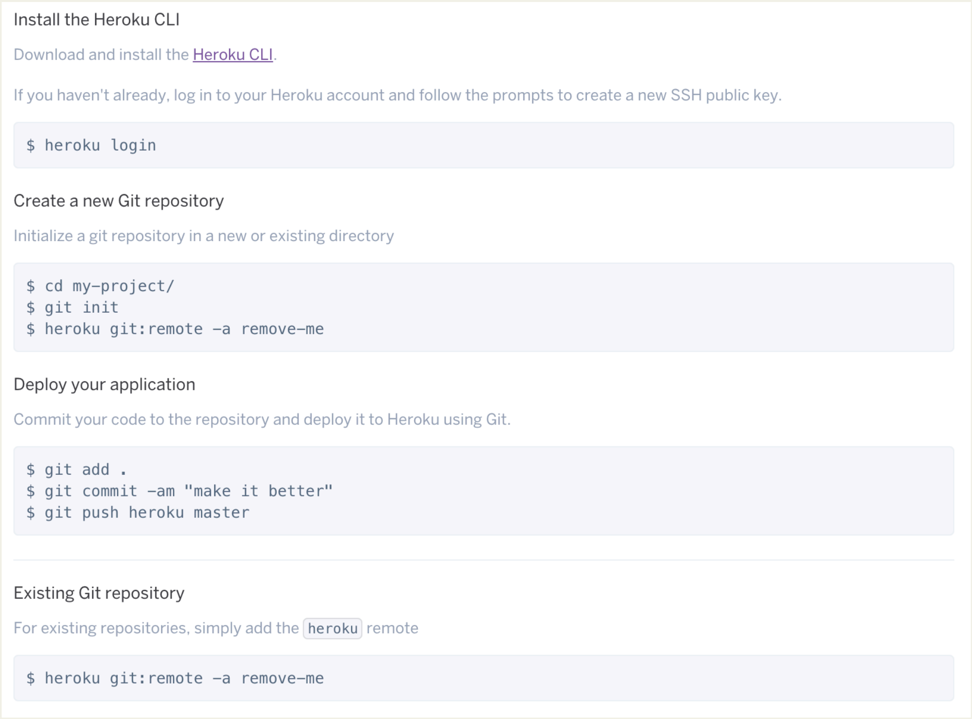
- Think about retrieving a single post.
- Why would you need it?
- What kind of information would you need to specify a single post?
    - How would your Post POJO change?
    - How would your URL schema change?
- Implement the corresponding functionality
- => Code

Solution >> Delete posts

- Add functionality to delete a post.
- What HTTP verb would you use?
- Implement it.
- => Code

Solution >> Deploy to heroku

- Create an account on http://www.heroku.com
- Follow the documentation to push and deploy your code to heroku
- Post a link at https://github.com/micromata/webengineering-2017/issues/3



Install the Heroku CLI

Download and install the Heroku CLI.

If you haven't already, log in to your Heroku account and follow the prompts to create a new SSH public key.

```
$ heroku login
```

Create a new Git repository

Initialize a git repository in a new or existing directory

```
$ cd my-project/
$ git init
$ heroku git:remote -a remove-me
```

Deploy your application

Commit your code to the repository and deploy it to Heroku using Git.

```
$ git add .
$ git commit -am "make it better"
$ git push heroku master
```

Existing Git repository

For existing repositories, simply add the `heroku` remote

```
$ heroku git:remote -a remove-me
```

Persistence >> the problem

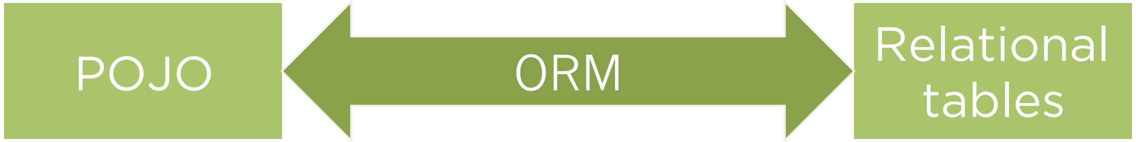
- Our application has a very simple in-memory data structure to store posts
- All data is lost on application restart
- Heroku might restart its deployments

- Solution: Use a „real database“

Persistence >> the challenge

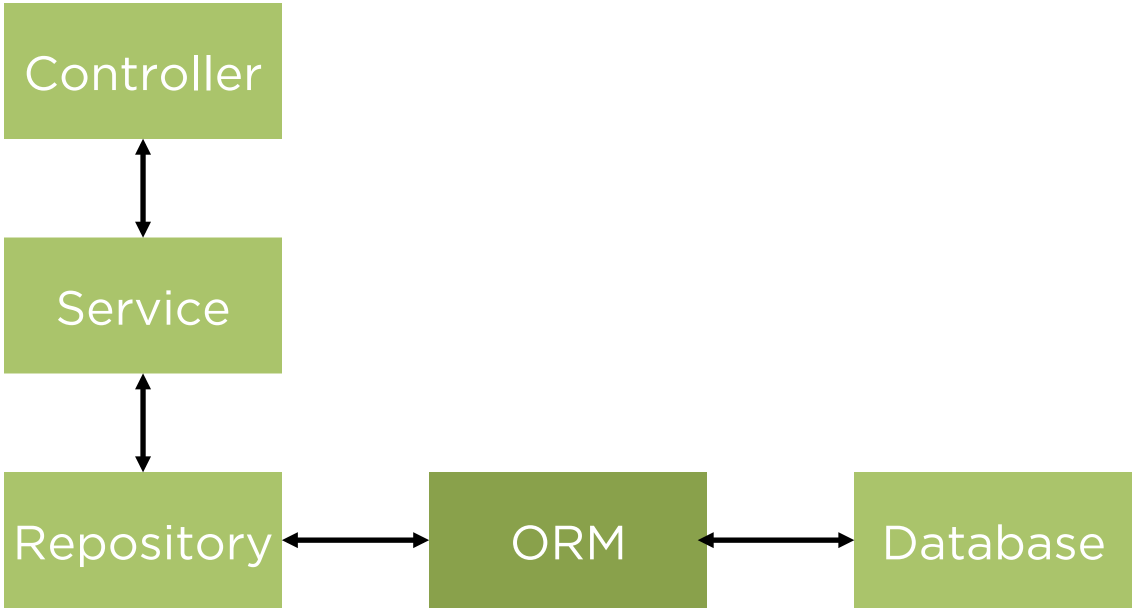
- Two worlds
  - POJOs and Java‘s object-oriented model
  - Relational tables

- Solution: ORM (Object-Relational-Mapper)
  - Framework to handle conversion of relations between these worlds **automatically**
  - **JPA is the standard**
  - **Hibernate & EclipseLink implement JPA**



Persistence >> architecture

- Controller
- Service
- Repository
- POJO
- ORM

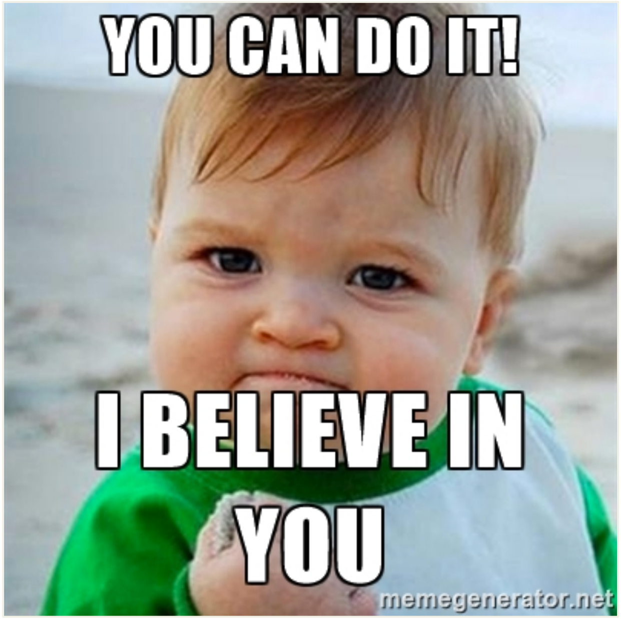


Exercises

Remark

- The exercises are not easy
- They train you to research, experiment, guess smart, …
- Talk to your fellow student

- Before I‘ll help you I usually ask what you have already tried. Be prepared, but not afraid :-).



Exercise >> Return Post-URL on post creation

- Return the URL to a single post on its creation as a JSON object with a field „url“

Exercise >> Automatic creationDate

- Is there a better way to fill the createdAt field?
- Hint: Look at JPA annotations!
- What advantages and disadvantages do both approaches have?

Exercise >> Persistent storage

- Currently we lose all data if the application shuts down
- Research H2 cheat sheet
- Configure your application such that a file-based H2 is used
- Store configuration at

```
src/main/resources/application.properties
```

- Test it

Exercise >> Check persistent storage

- Use the H2 jar to start its web-based console
- Examine the file-based database and the table structure of the table POST
- Hint: Stop your application

- Optional
  - Any idea to prevent having to use the hint (H2 Cheat sheet)?

Exercise >> Posts ordered by creation date

- Create a method in the CrudRepository to retrieve posts ordered by their date
- Hint: JQL, Query-annotation

Exercise >> Title length?

- What happens when you try to save a title with more than 255 chars?
- Why?
- What can you do to improve this?
    - Hint: annotations
- What can you do to prevent this?
- Which approach is better? Why?

Exercise >> Use Postgres on Heroku

- Research: why can‘t we simply use H2 for persistent storage on heroku?
- Optional: locally use postgres instead of H2
  - (Install database, create users, …)
  - Configure a separate application-profile for heroku deployment
- Adapt Procfile to use it
- Configure heroku to access a free Postgres instance
- Test it by manually restarting your heroku instance

This is **extremly difficult** – I do not expect that you can solve this; just for these people who are already done with everything and/or are really bored. All steps will be explained with a lot of details in the next lecture.