Webengineering

2017-05-15

Dr. Michael Lesniak

mlesniak@micromata.de

Roadmap 2017-08-05

- Response to feedback
- Best practice: TODO
- Landscape
- Solution to exercises

- Postgres with Heroku
- Authentication
  - User POJO
  - Attach User to Post
  - Authenticate over JWT

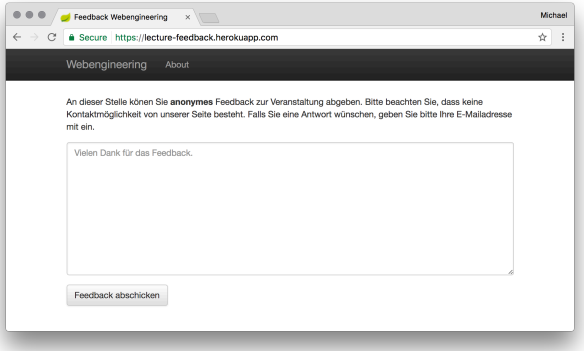
- Exercises

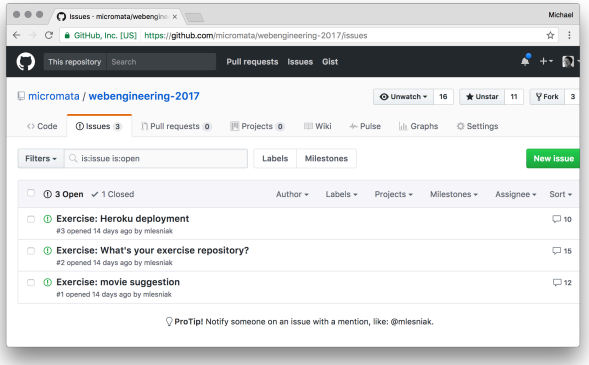
Response to feedback

- Video quality will increase to 720p
- Remove auto correct from slides
- Post topics of the lecture beforehand (is sunday enough)?

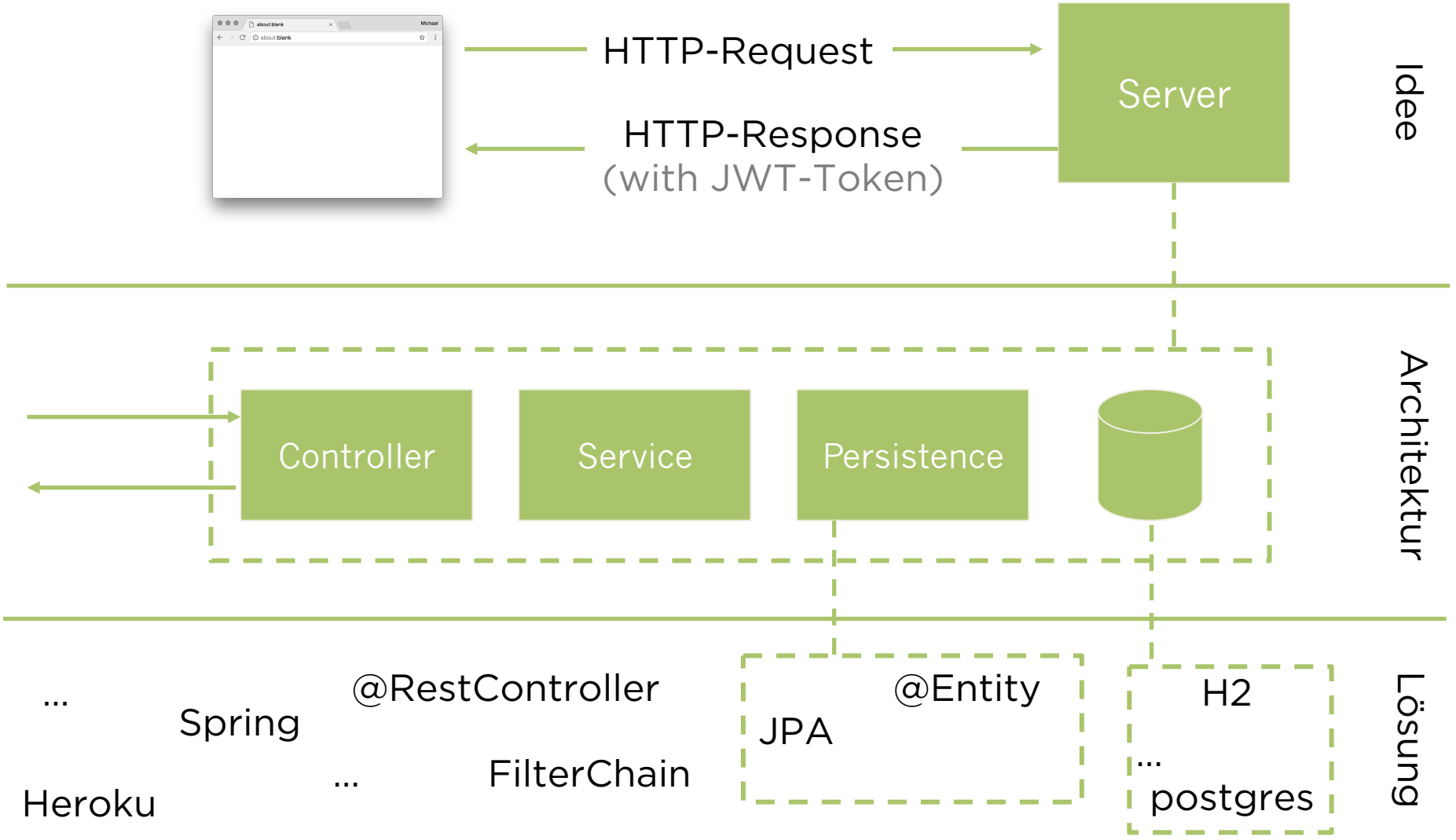
- Topics for last third of lecture
  - JSON-HAL
  - Websockets
  - … you can decide (or I will)

- **Questions to answer for next time**
  - **Difficulty & Speed – ok?**
  - **Style of lecture?**
  - **Ratio of theory vs praxis?**





Concepts vs. Architecture vs. Implementation

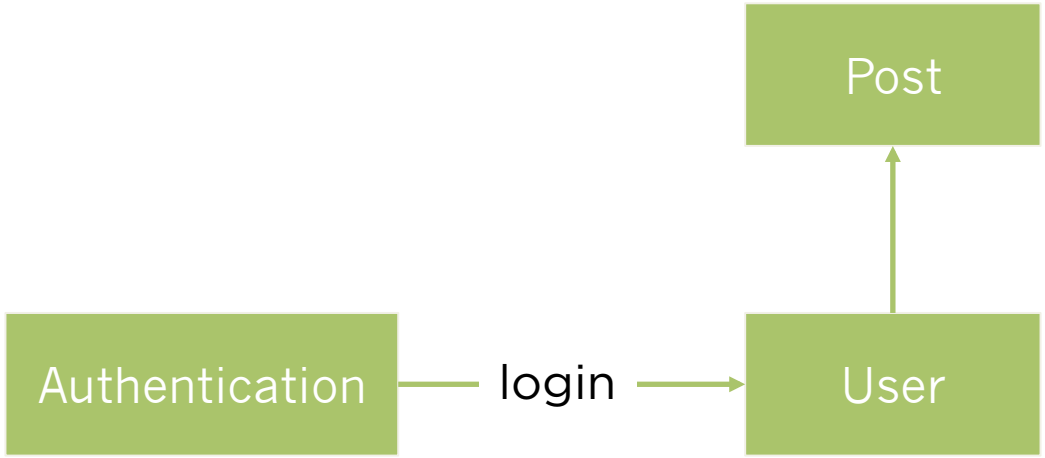


Persistence on Heroku

- H2 uses files to store its tables
- Heroku does not allow to store files (PaaS)
- Solution: Use a supported database
- Steps
  - Enable database support for your application
  - Create dedicated configuration file for heroku
  - Configure Spring to use the configuration file on heroku
  - Redeploy
- Test it

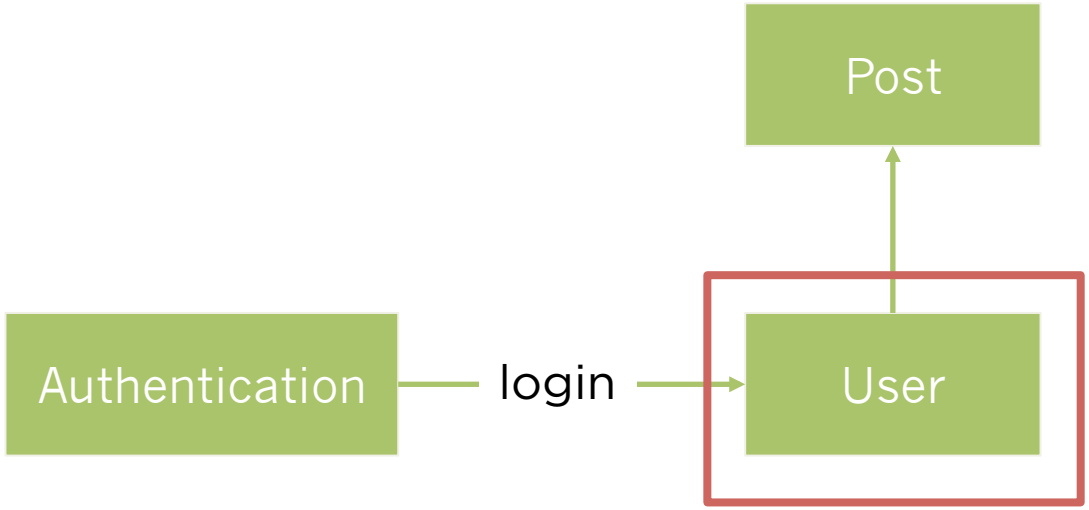
Authentication >> idea and concept

- Not every user should be allowed to perform every operation
- Examples
  - Only logged-in users should perform operation Foo
  - Users should only be able to modify their own entities
  - ...



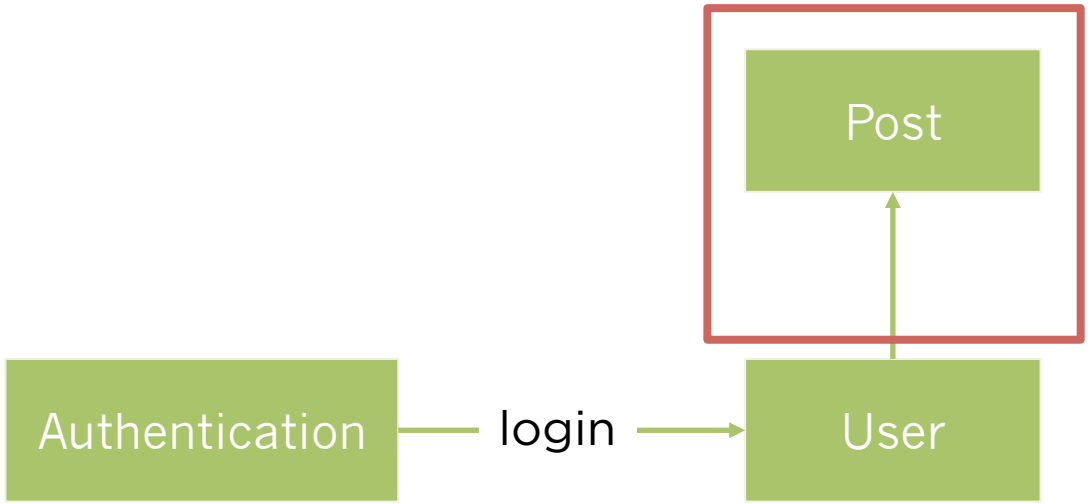
Authentication >> User POJO

- POJO
  - User entity
- Repository
  - Annotations
  - UserRepository



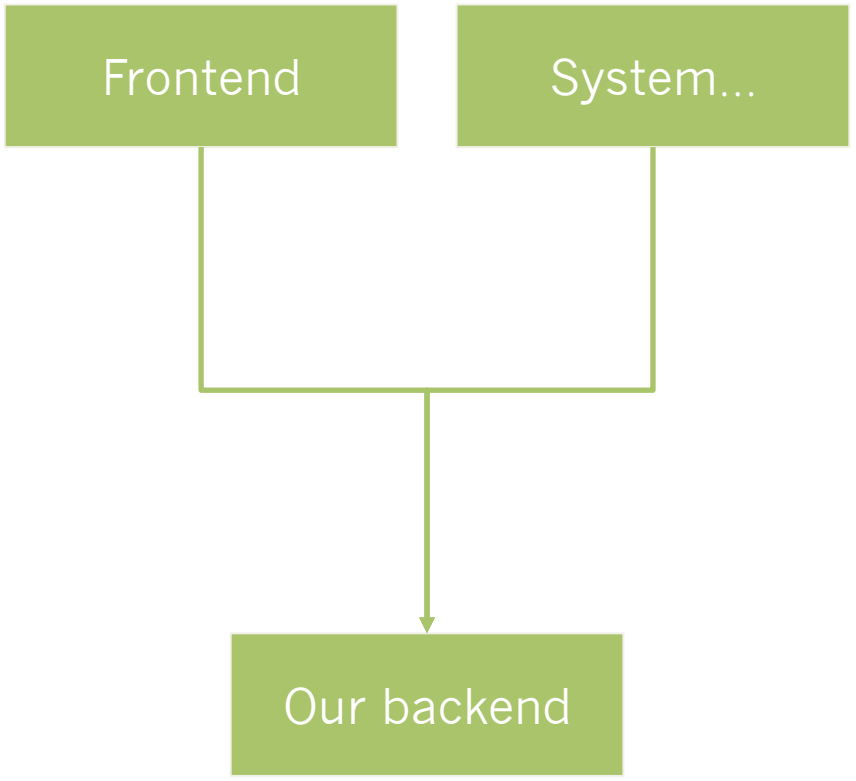
Authentication >> Attach user to post

- Attach a required user to each post
- @ManyToOne
  - optional = false
- UserService to retrieve current user

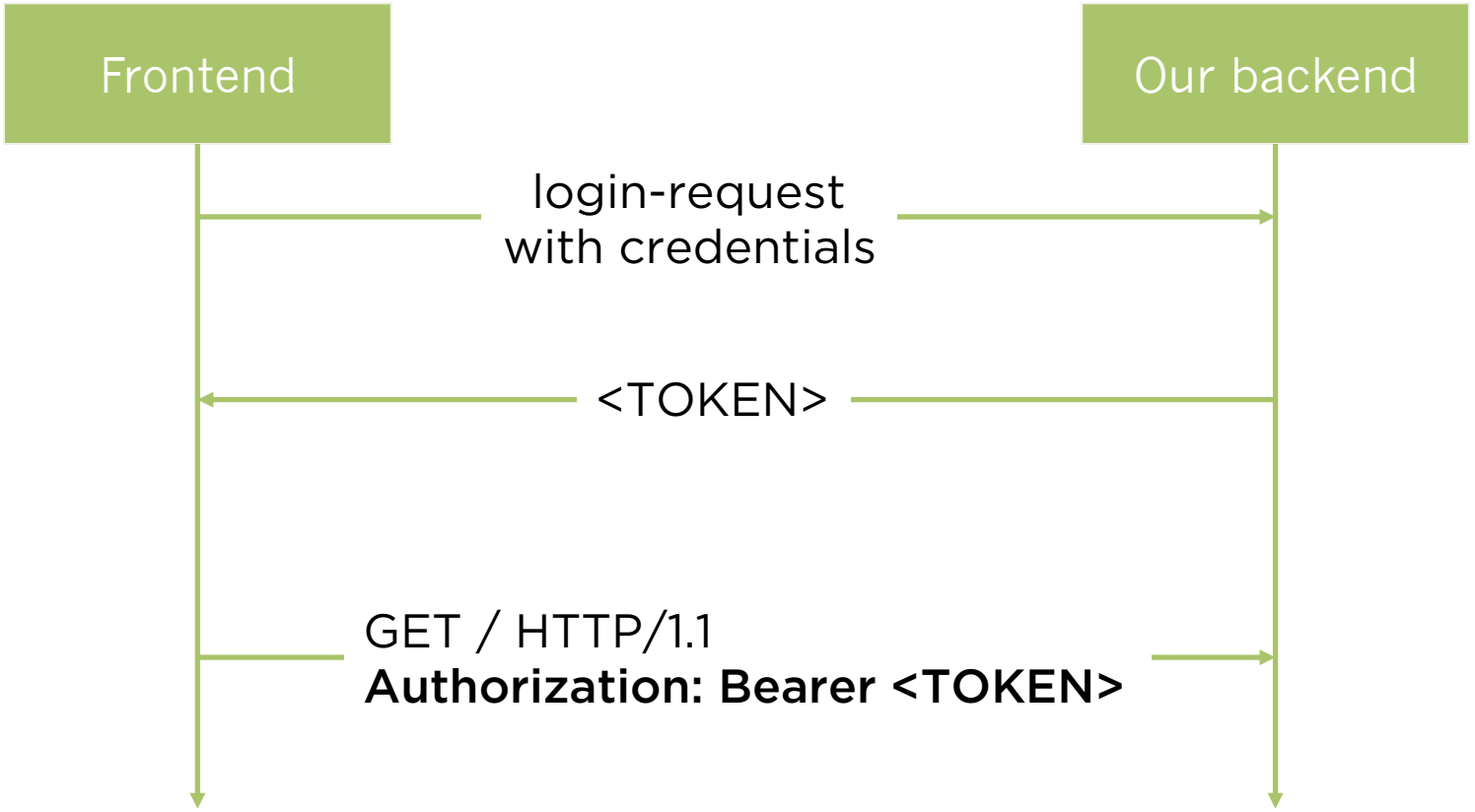


Authentication >> problem

- Strict separation between frontend and backend
- In addition: backends which are not based on Browsers
- We still want to authorized users
- Solution: Token-based requests

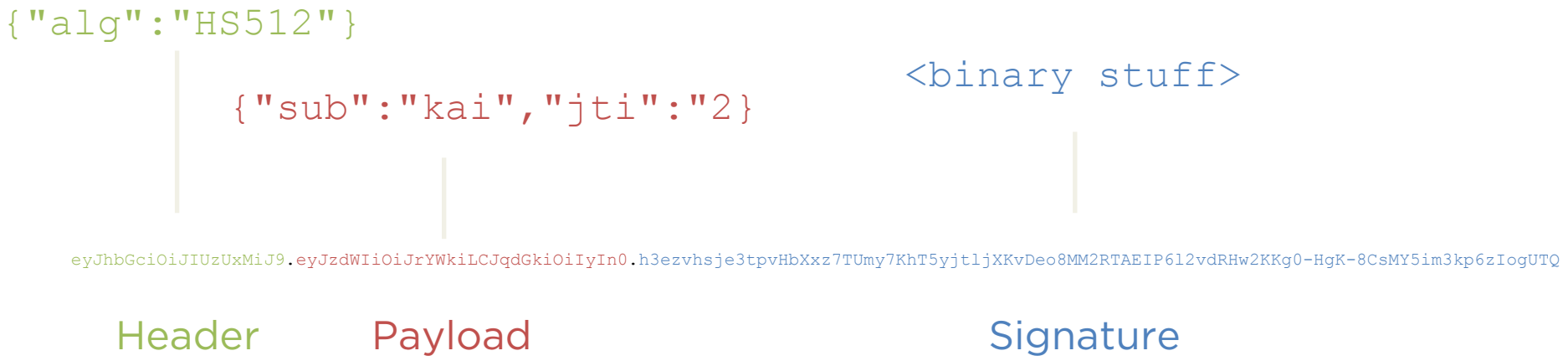


Authentication >> solution // token-based auth.



Authentication >> JWT: a modern solution

- JSON Web Tokens
- Standardized
- Structure
  - Three parts
  - Base64-encoded



Reminder

- The concepts are general
  - tokens
  - Authorization-Header in HTTP requests
- We will now implement a lot of things which are **framework-specific**, e.g.
  - Filters
  - Spring-Security Context-Holder
  - ...

Exercises

Answer the following questions…

- Difficulty & Speed – ok?
- Style of lecture?
- Ratio of theory vs Reality?

Use postgres on your Heroku instance

- Configure your instance to use Postgres
- Add application profile
- Deploy it
- Test it (hint: restart heroku instance)

How to handle POST-Create URL host handling?

- Create a post on your heroku instance
- What is the returned URL?
- Optional: Try to find out / think about why
- Think about different approaches to fix this
  - Advantages?
  - Disadvantages?

Password-Hashing

- SALT-ing of passwords
  - What is it?
  - Why should you use it?
- Add a simple salting mechanism to password checking

Add CREATED_AT in data-h2.sql

- Add `createdAt`-information in data-h2.sql
- Hints
  - Look at schema in h2 table
  - H2 has conversion functions

Secret in AuthService

- The secret in AuthService is fixed
- What can you do to change it?
- Hint: @Value annotation

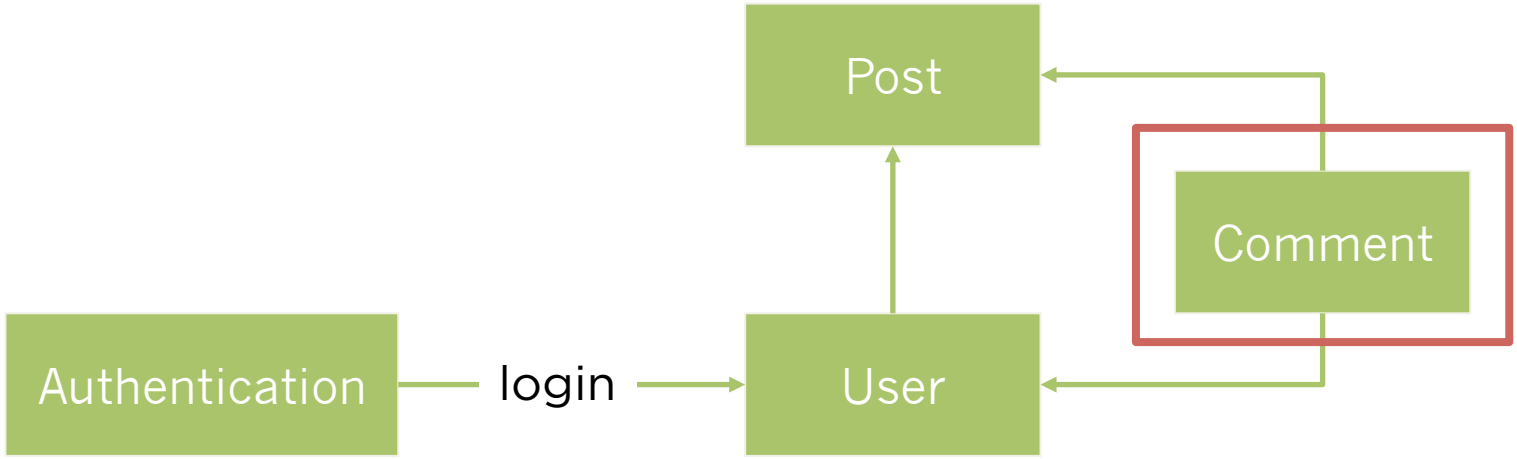
Add Logging

- Look at my examples in Code
- Why is logging relevant?
- Distribute functional logging in services

;-)

Add Comments to post

- Add all necessary functionality to add comments to an existing (or new) post
  - Controller
  - Service
  - Repository
  - Add comments to posts (Hint @OneToMany)



Note that this is a lot of work, but 90% has already been shown in the lecture