Studiengangsabend: Informatik
Live-Coding @ HSR

Martin Kempf, Reto Kleeb

September 2015

ELCA *We make it work.*

ELCA

WHO ARE WE?

Martin Kempf

Senior Software Engineer

At ELCA since 2013

BSc in Computer Science: Hochschule f. Technik Rapperswil 2008

MSc in Computer Science: Hochschule f. Technik Rapperswil 2012

Reto Kleeb

Senior Software Engineer

At ELCA since 2012

BSc in Computer Science: Hochschule f. Technik Rapperswil 2008

MSc in Computer Science: Northeastern University 2012

GAME IDEA

- Competitive / Concurrent Solving of a Shared Sudoku Puzzle:

  - If a participant enters a correct value, this value is shared with all other participants (field becomes read-only)

  - Correct Values are rewarded with +6 Points

  - Invalid Values are punished with -2 Points

- The game is over as soon as there are no empty fields left.

- Based on SE2-Project

- Sudoku Battle Royal (V1)

  - Swing

  - Manual Socket-Handling / Protocol

  - Year ~2007…

ELCA

GAME IDEA

- Sudoku Battle Royal (V2):
  - Browser Based Client (JS)
  - Communication trough Websockets
  - Simple Webserver based on Spring Boot

Sudoku Battle Royal 2

Your Position

1. reto.kleeb@elca.ch — **10 Pkt.**

Ranking

1. reto.kleeb@elca.ch — **10 Pkt.**
2. martin.kempf@elca.ch — **0 Pkt.**

Game

```
      1 8
4 5 6       8 9 1
8 3 2 5   1 7 6
        6 8
5 6 4           8
  2         9 7 6
        8 5 6 1 2
6 1   3 2 4 5   9
  8   6   9 3 4
```

Action Log

```
reto.kleeb@elca.ch: Scored: 10
Application: Game Started
reto.kleeb@elca.ch: Registered
```

ELCA

SOLUTION: MESSAGING OVER WEBSOCKETS

- Websockets

  - Standing Socket between Browser und Webserver

  - Duplex Communication

  - Supported in all modern Browsers and Servers

 - Allows simple server to client notifications (as opposed to classic HTTP)

- Our Solution (Messaging)

  - Use a message-based protocol on top of the websocket

  - Messaging conventions/protocol simplify client/server communication

  - Broker provides support for topic registrations and redistribution
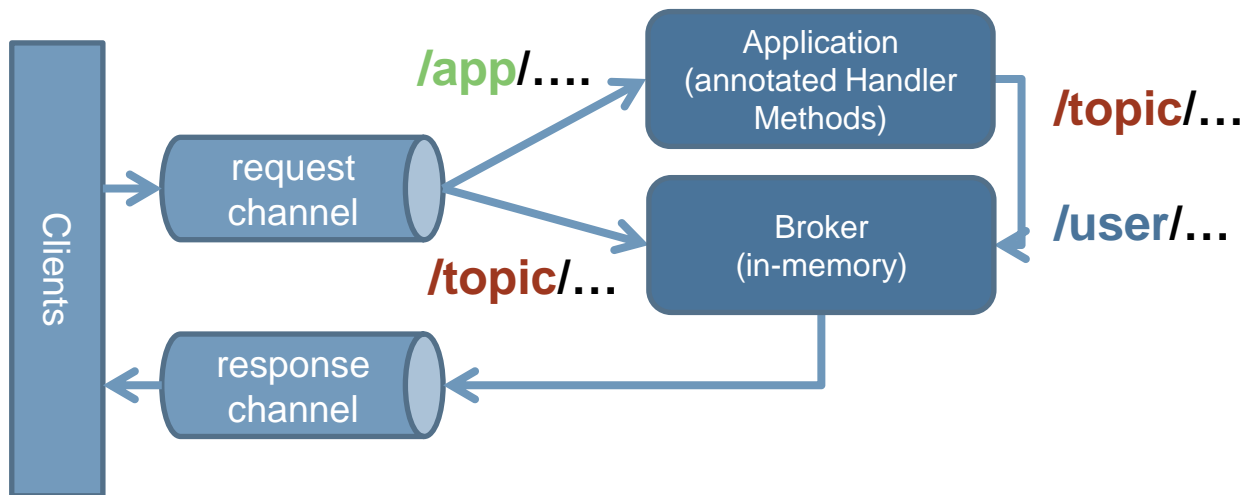
ELCA

MESSAGING: BUILDING BLOCKS

- Subscribe to a topic
  - Public topics
  - Personal topics
- Send Messages to a destination
- On top of Websocket "/socksjsendpoint"
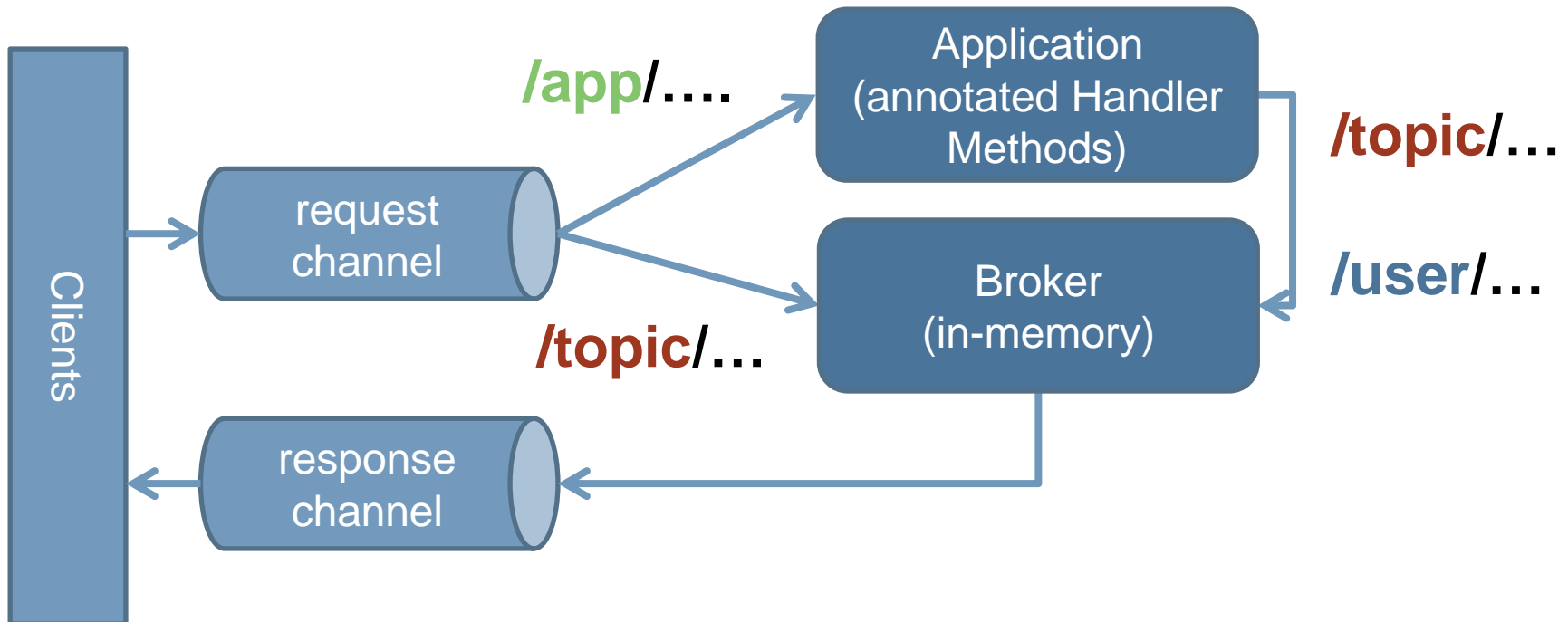
MESSAGING: BUILDING BLOCKS

```
>>> SUBSCRIBE
id:sub-0
destination:/topic/game/login
```

```
<<< MESSAGE
destination:/topic/game/login
content-type:application/json;charset=UTF-8
subscription:sub-0
message-id:n_tlbiyq-1
content-length:37

{"playerName":"martin.kempf@elca.ch"}
```

Clients

request channel

/app/....

Application
(annotated Handler Methods)

/topic/…

Broker
(in-memory)

/topic/…

/user/…

response channel

ELCA

TECHNOLOGY: JAVASCRIPT / WEBSOCKET SUPPORT

sock.js

- SockJS provides a "WebSocket-like object"

- Consistent Socket API in all Browsers and network environments

- https://github.com/sockjs/sockjs-client

stomp.js

- STOMP: Simple (or Streaming) Text Orientated Messaging Protocol

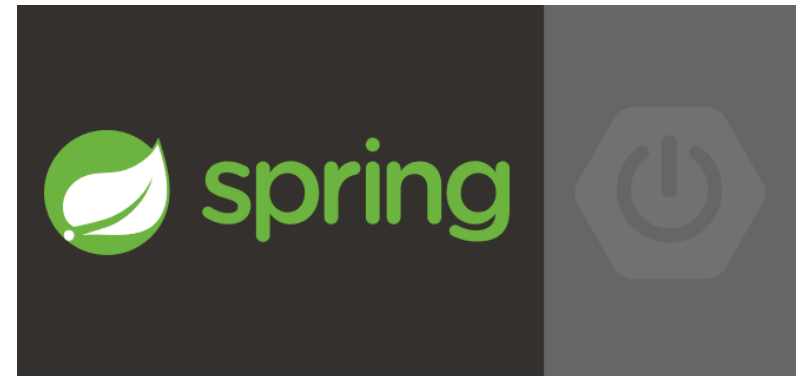- stomp.js allows us to easily use STOMP on top of a Websocket

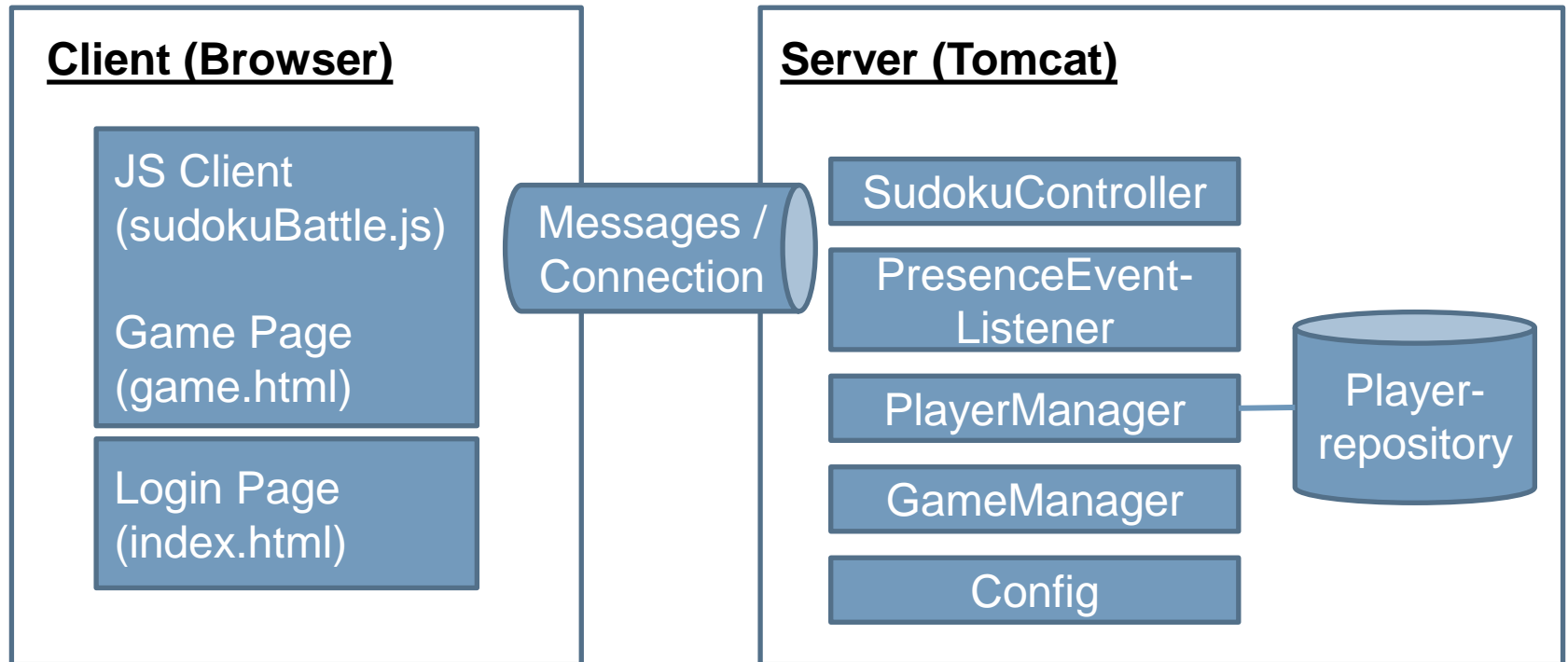- http://jmesnil.net/stomp-websocket/doc/

vue.js

- Links View and Model via two way data bindings (similar to the AngularJS bindings)

- DOM manipulations are abstracted away

- http://vuejs.org

TECHNOLOGY: SPRING BOOT

- Spring (Boot)

  - Dependency Injection

  - Simple REST Endpoints

  - Spring Security for authentication

  - Simple In-Memory Persistence out of the Box (Spring Data JPA)

  - In-Memory Broker for STOMP Messages

APPLICATION OVERVIEW

Client (Browser)

JS Client
(sudokuBattle.js)

Game Page
(game.html)

Login Page
(index.html)

Messages /
Connection

Server (Tomcat)

SudokuController

PresenceEvent-
Listener

PlayerManager

GameManager

Config

Player-
repository

STAGE 0: BASIC SETUP

Client (Browser)

JS Client
(sudokuBattle.js)

Game Page
(game.html)

Login Page
(index.html)

Messages /
Connection

Server (Tomcat)

SudokuController

PresenceEvent-
Listener

PlayerManager

GameManager

Config

Player-
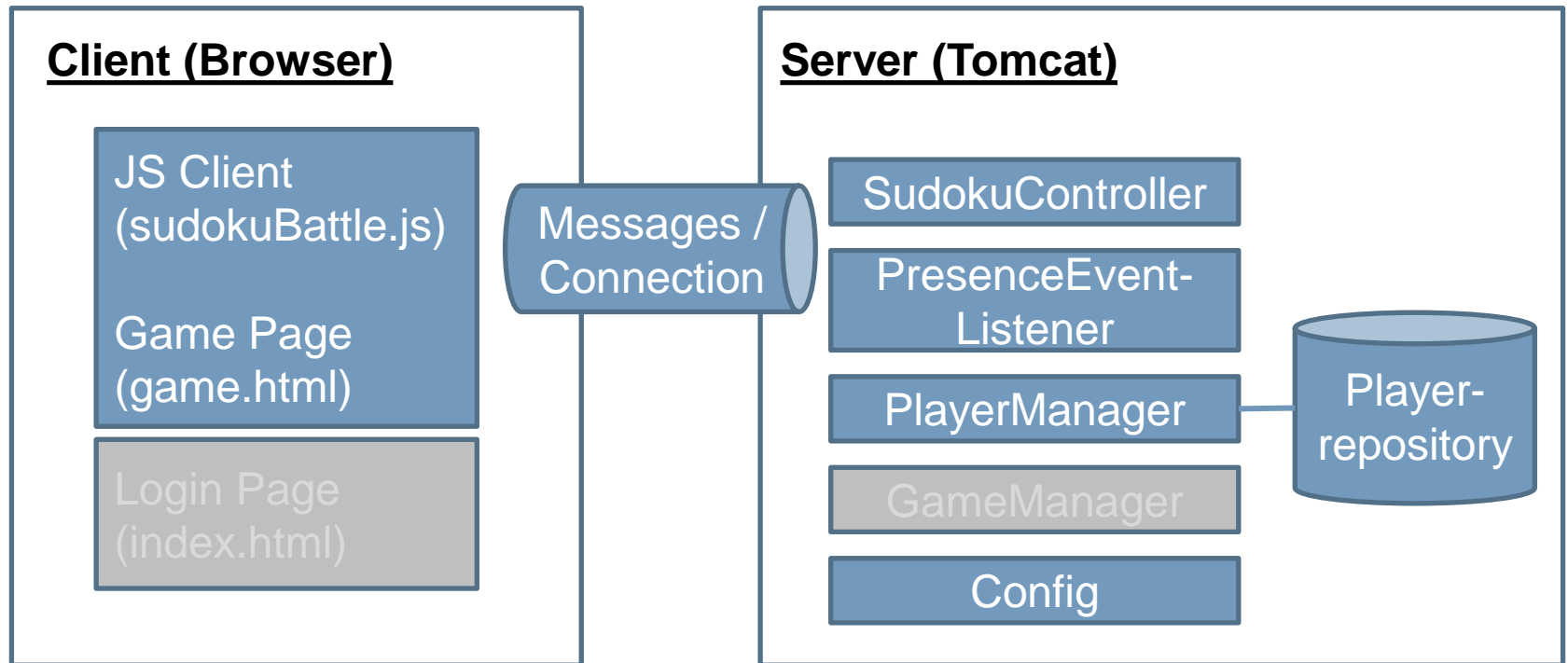repository

- Plain Spring Boot Scaffold that displays static HTML page

- Basic Spring Security Config

- Empty Controller for incoming messages

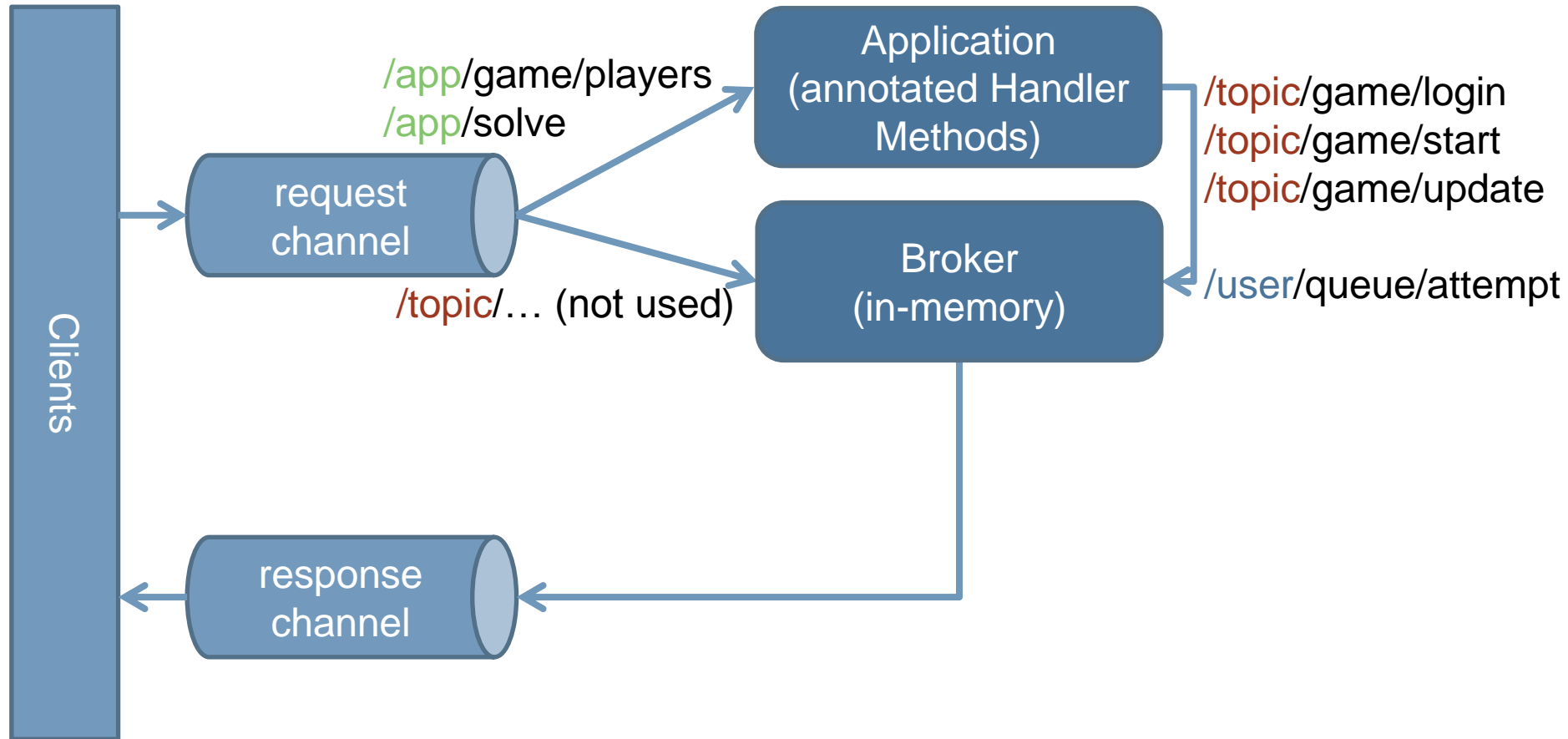- Repository to persist Players

ELCA
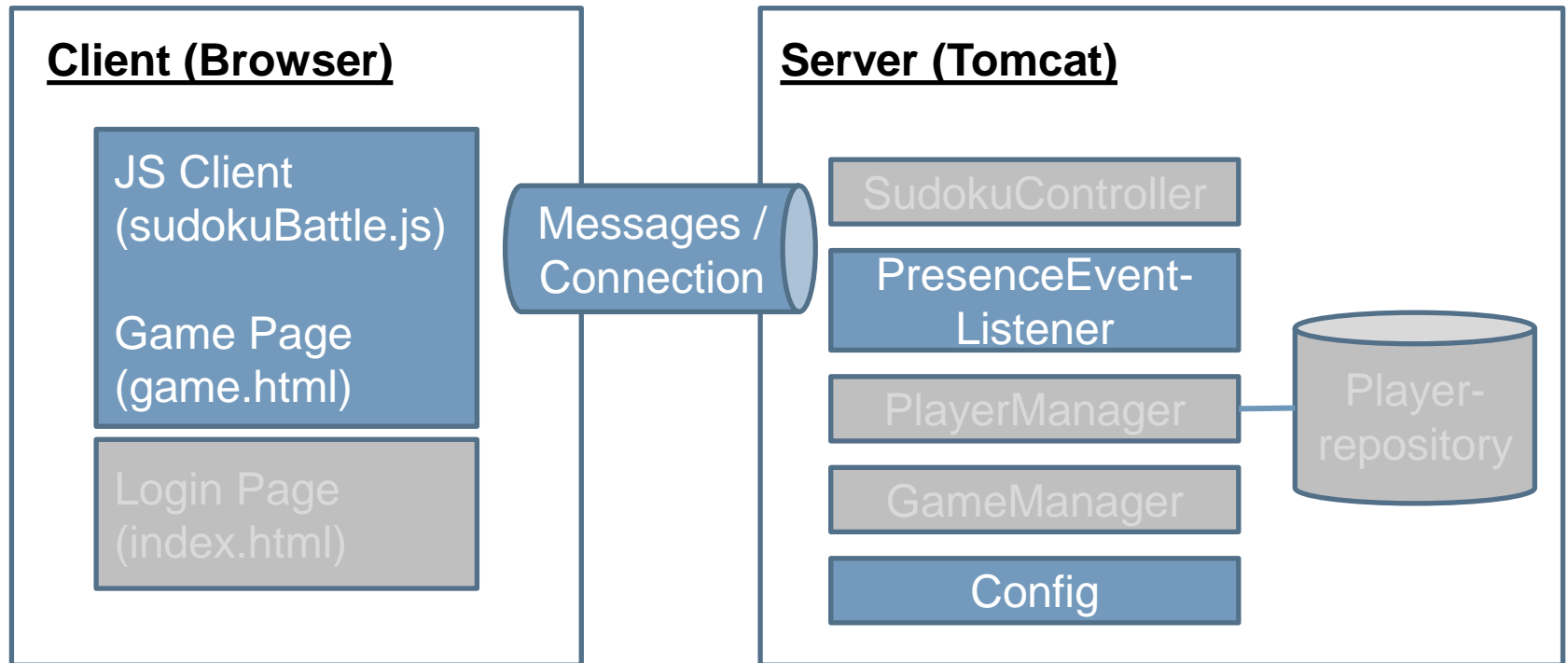
STAGE 1: SOCKET SETUP

- Setup of Socket-Endpoint and Message-Broker

- Handling of "Session Connect Event" (stores Players in the Repo)

- Return a list of all (already) connected players on connect

- Client: Connect to Socket and subscribe to destination

ELCA

MESSAGING: BUILDING BLOCKS

■ On top of Websocket "/socksjsendpoint"

Clients

/app/game/players
/app/solve

request channel

/topic/… (not used)

Application (annotated Handler Methods)

Broker (in-memory)

/topic/game/login
/topic/game/start
/topic/game/update

/user/queue/attempt

response channel

ELCA

STAGE 2: SENDING MESSAGES FROM THE SERVER

Client (Browser)

JS Client
(sudokuBattle.js)

Game Page
(game.html)

Login Page
(index.html)

Messages /
Connection

Server (Tomcat)

SudokuController

PresenceEvent-
Listener

PlayerManager

GameManager
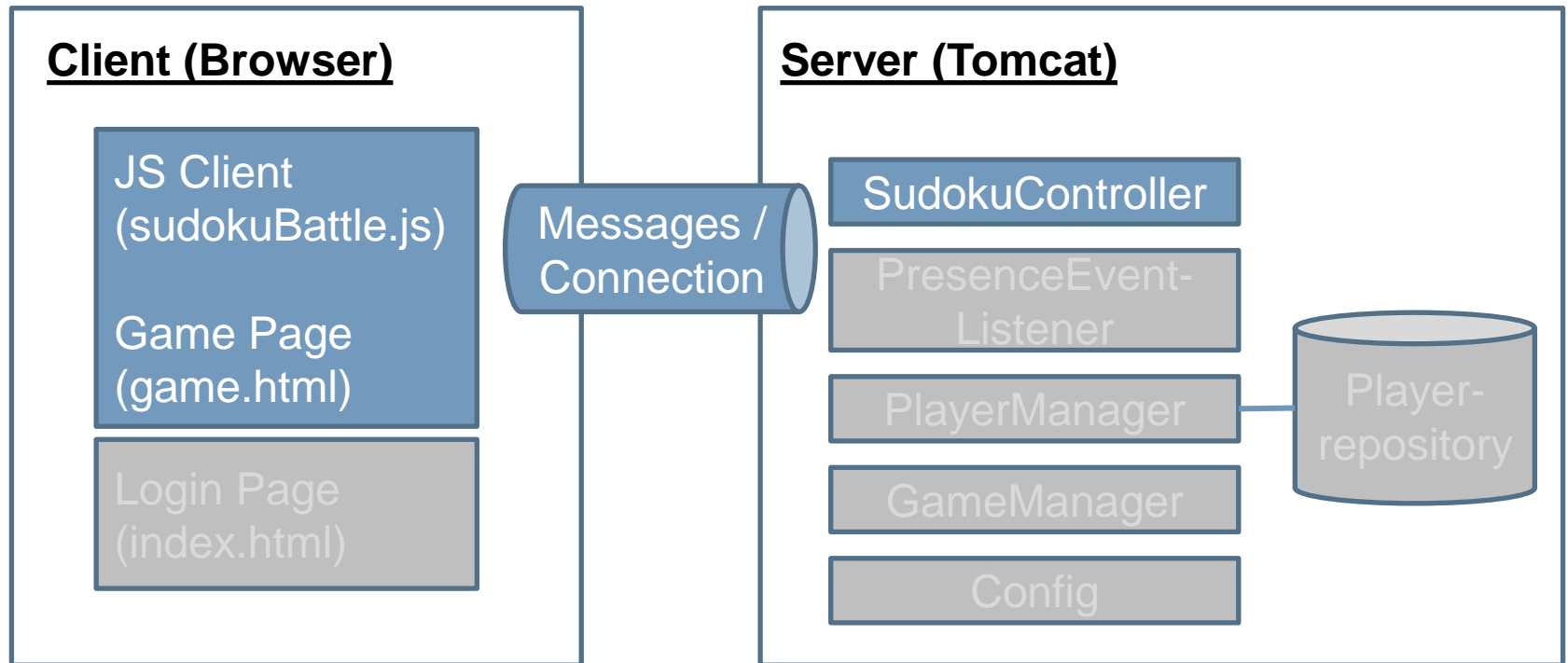
Config

Player-
repository

- Setup of Broker for "/topic" prefixed destinations
- Sending of messages to "/topic/game/login" after new session connections
- Client: Subscription to destination "/topic/game/login"

MESSAGING: BUILDING BLOCKS

lists the players that are already waiting

/app/game/players
/app/solve

inform all waiting players about a new player

/topic/game/login

send a "solve attempt" from client to server

inform all players about a board /score update

/topic/game/start

inform all players about the start of the game

/topic/game/update

/user/queue/attempt

inform a particular user about the outcome of an attempt

Clients

Application (annotated Handler
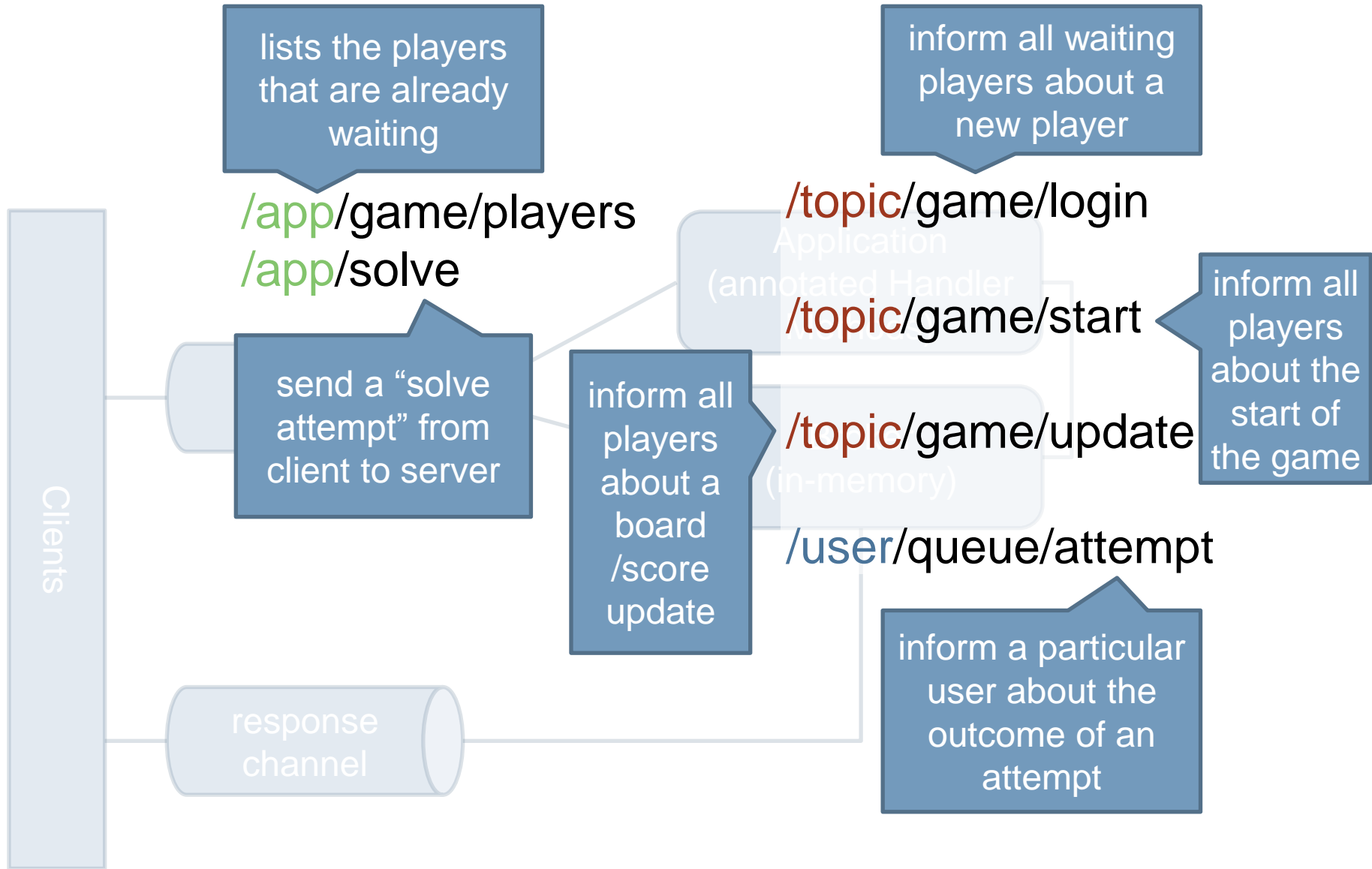
(in-memory)

response channel

ELCA

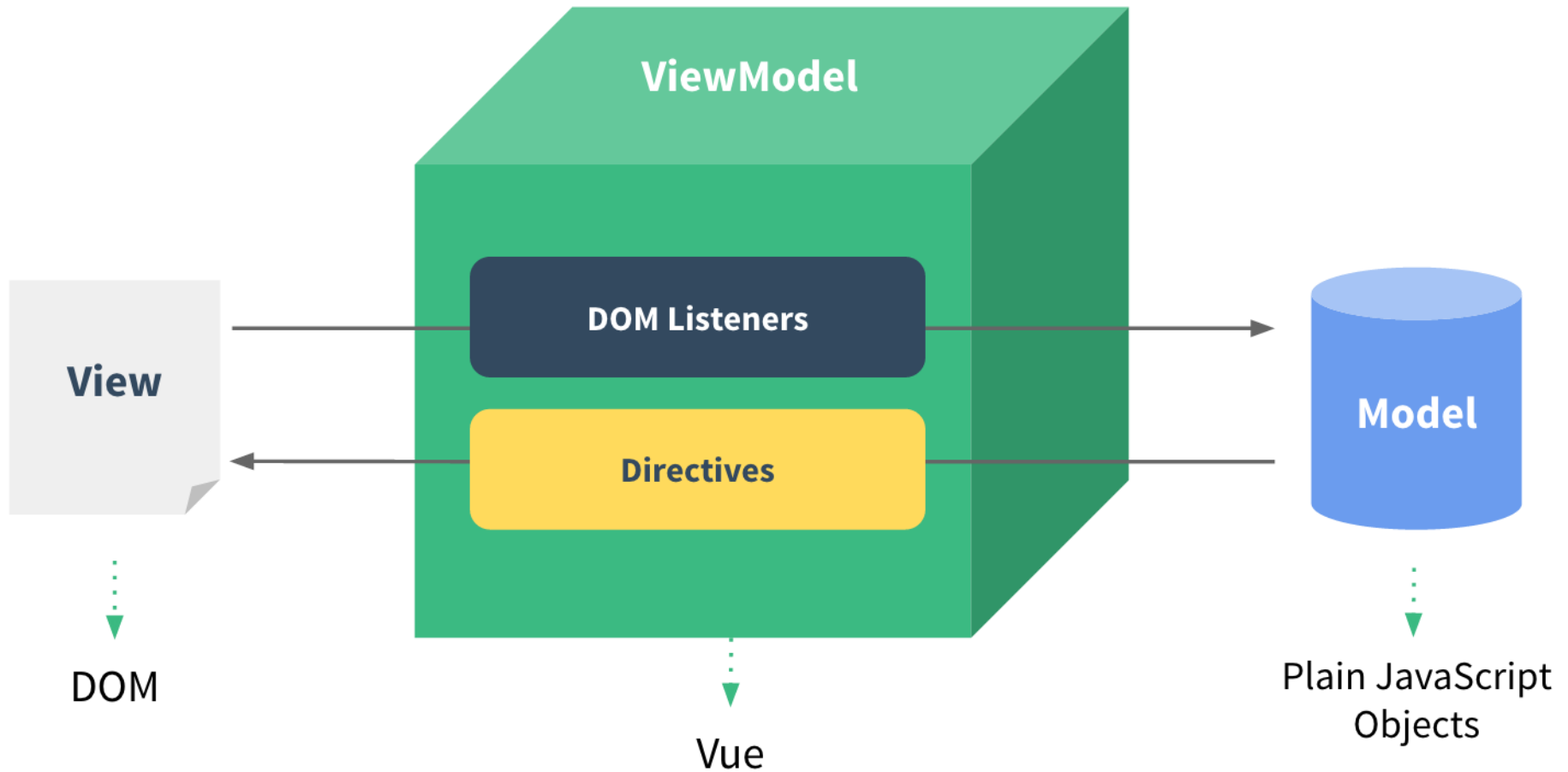# STAGE 3: SENDING MESSAGES FROM THE CLIENT

- ■ Handling of "Solve" Messages on the Server
    - ▪ Messages to public destination
    - ▪ Messages to personal destination
- ■ Sending "Solve Messages" from the Client

MESSAGING: BUILDING BLOCKS

lists the players that are already waiting

/app/game/players
/app/solve

inform all waiting players about a new player

/topic/game/login

send a "solve attempt" from client to server

inform all players about a board /score update

/topic/game/start

inform all players about the start of the game

/topic/game/update

/user/queue/attempt

inform a particular user about the outcome of an attempt

Clients

Application (annotated Handler

(in-memory)

response channel

ELCA

- Compilation and Packaging (using Maven)

- Upload of the JAR file to a PAAS Provider
  (using the «cf» command line tool)

THE "OFFICIAL" RULES

- All participating players are divided into 3-5 player-groups for the 3-5 elimination rounds

- All participating players *(of the current elimination round)* log in (using their email address)

- The administrator starts the game

- Once a game has been started, no new players can connect

- Once the puzzle has been completely solved the game is over

- The top 2 players of each elimination round advance to the final round

- The player with the top score in the final round wins the price

- In case of a tie we will draw a winner

■ **Signup (FIRST): http://bit.do/hsr**

■ **Play: http://battletest.cfapps.io/**

URLS

- Github Repository: https://github.com/ret0/sudokuBattleRoyal2

  - Contains the complete sources and these slides

- Related Examples and Tutorials:

  - https://spring.io/guides/gs/messaging-stomp-websocket/

  - https://github.com/salmar/spring-websocket-chat

  - http://g00glen00b.be/spring-angular-sockjs/

ELCA

Thank you.

Contact

Martin Kempf	Reto Kleeb
Senior Software Engineer	Senior Software Engineer
martin.kempf@elca.ch	reto.kleeb@elca.ch

ELCA Informatique SA | Lausanne 021 613 21 11 | Genève 022 307 15 11
ELCA Informatik AG | Zürich 044 456 32 11 | Bern 031 556 63 11

www.elca.ch

ELCA