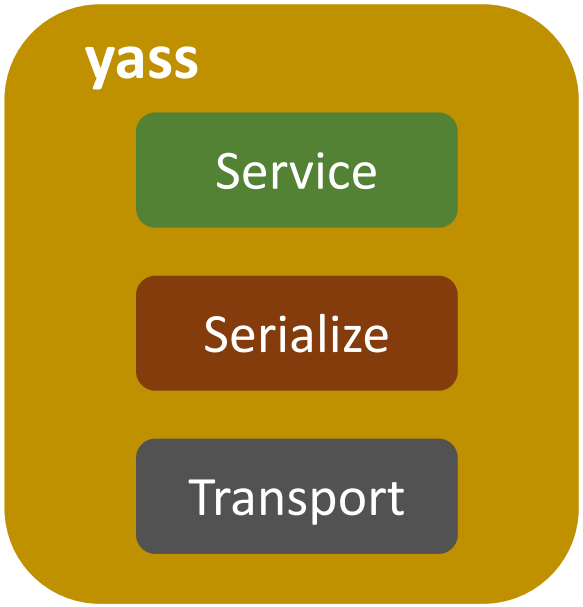
yass ? Yet Another Service Solution

- a small software library for efficient peer-to-peer communication (FastSerializer)
  - Java (3500 LOC, 150KB jar)
  - TypeScript (900 LOC)
  - Python 2 & 3 (with support for type hints, 700 LOC)
  - high throughput, low latency, reactive services
- explicit type-safe contract with DTOs and interfaces
- session based, bidirectional message streaming (sync/async, oneway/rpc)
- Open Source (BSD-style license)
  - https://github.com/softappeal/yass
  - Maven Central: groupId=ch.softappeal.yass

Design



- maps contract (DTOs and interfaces) to messages
- transforms messages to byte chunks
- transports byte chunks between (distributed) processes
  - TCP/IP socket (650 LOC)
    - SSL/TLS
  - WebSocket (200 LOC)
    - “socket over http”
    - Java (JSR 356, Java API for WebSocket)
    - TypeScript (Browser WebSocket)

Service ?

service contract (interface)

```
interface Calculator {
  int add     (int a, int b);
  int multiply(int a, int b);
}
```

service implementation (server side)

```
class CalculatorImpl implements Calculator {
  int add     (int a, int b) { return a + b; }
  int multiply(int a, int b) { return a * b; }
}
```

service usage (client side)

```
Calculator calculator = new CalculatorImpl();
int a = calculator.add(1, 2);
int m = calculator.multiply(2, 3);
```

Interceptor (AOP, around advice)

printing calculator

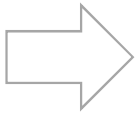
```
class Printer implements Calculator {
  int add(int a, int b) {
    System.out.println("add");
    return a + b;
  }
  int multiply(int a, int b) {
    System.out.println("multiply");
    return a * b;
  }
}
```

logging interceptor

```
Interceptor LOGGER =
(method, arguments, invocation) -> {
  System.out.println(method.getName());
  return invocation.proceed();
};
```

usage

```
Calculator calculator = Interceptor.proxy(
  Calculator.class,
  new CalculatorImpl(),
  LOGGER // , PROFILER, AUTHORIZATOR
);
int a = calculator.add(1, 2);
int m = calculator.multiply(2, 3));
```



output

```
add
multiply
```

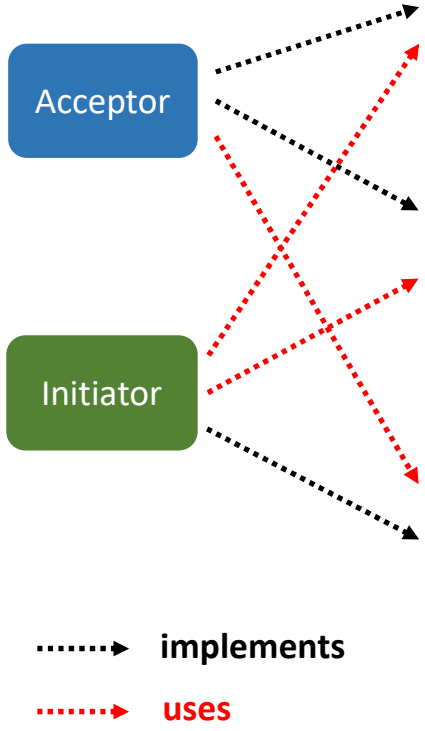
Contract

DTOs

```
class Stock {
  int id;
  String name;
  Rating rating;
}
```

```
enum Rating { AAA, AA, A }
```

```
class Price {
  int stockId;
  int bid;
  int ask;
}
```



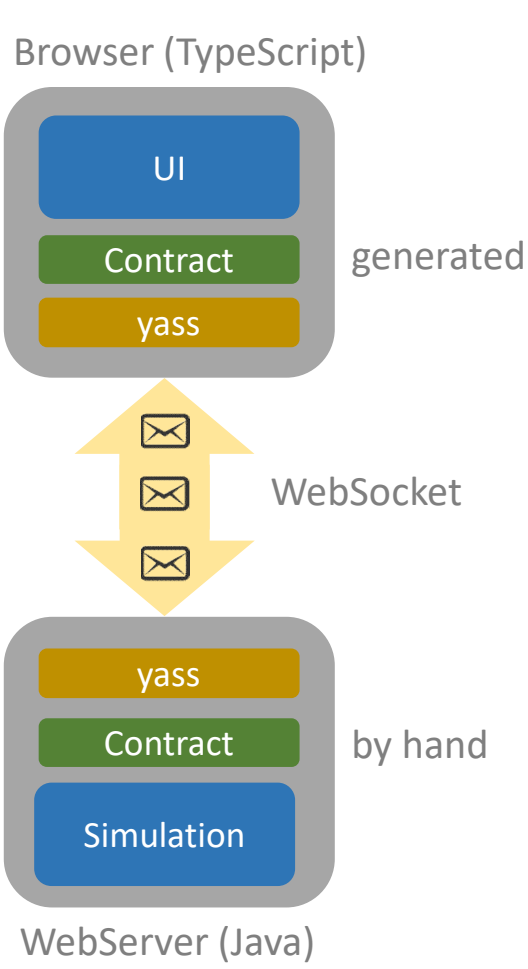
interfaces

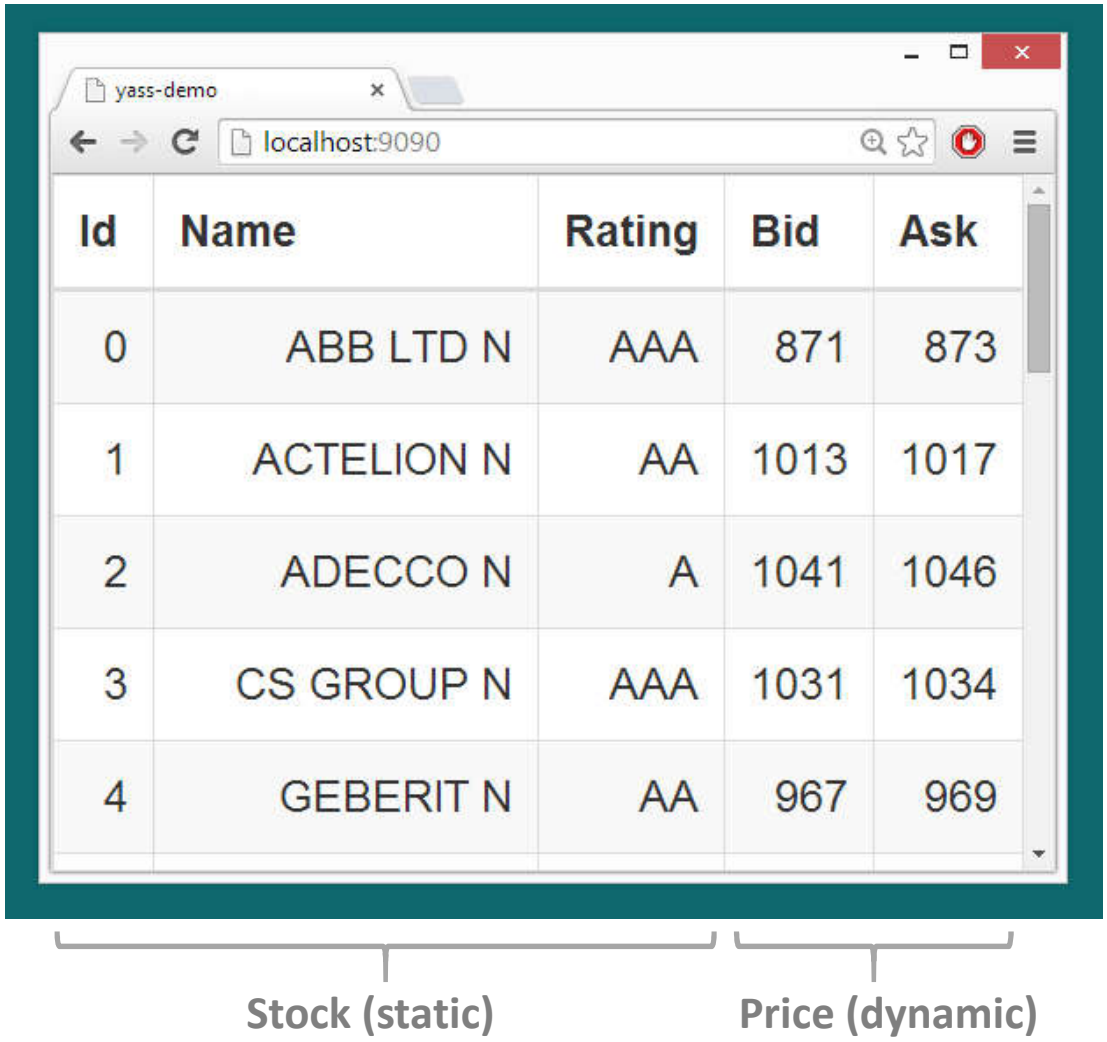
```
interface StockService {
  List<Stock> getStocks();
}
```

```
interface PriceService {
  void subscribe  (List<Integer> stockIds);
  void unsubscribe(List<Integer> stockIds);
}
```

```
interface PriceListener {
  @OneWay void notify(List<Price> prices);
}
```

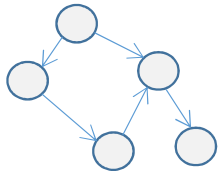
Tutorial


Browser (TypeScript)
UI
Contract
yass
generated
WebSocket
yass
Contract
Simulation
by hand
WebServer (Java)


yass-demo
localhost:9090
Id
Name
Rating
Bid
Ask
0
ABB LTD N
AAA
871
873
1
ACTELION N
AA
1013
1017
2
ADECCO N
A
1041
1046
3
CS GROUP N
AAA
1031
1034
4
GEBERIT N
AA
967
969
Stock (static)
Price (dynamic)

Serialize

message (graph in process memory)



```
Stock stock = new Stock();
stock.id = 31;
stock.name = "ABB";
stock.rating = Rating.AAA;
```

transforms messages to byte chunks

XML serializer

```
<Stock>
  <id>31</id>
  <name>ABB</name>
  <Rating>AAA</Rating>
</Stock>
```

78 bytes

FastSerializer (1000 LOC)

```
01 01 1F 02 03 41 42 42 03 00 00
```

- Tags, Base 128 Varints
- UTF-8 strings
- Optional fields
- Custom BaseTypes

11 bytes

Dumper

```
Stock(
  id = 31
  name = "ABB"
  rating = AAA
)
```