Running a real factory on WebSockets

Joakim Kemeny
@joakimkemeny

CALLISTA
Enterprise

What is a real factory?

What is a real factory?

Hardware

Scales

Silos

Conveyor Belts

Mixers

Boilers

Coffee Makers

Pipes

What is a real factory?

Cranes

Hardware

Tambours

Engines

I/O Units

PLCs

Winders

Lathes

What is a real factory?

Hardware

I/O Units PLCs

Software

What is a real factory?

Hardware

I/O Units PLCs

Manage orders

List orders, Add order, Cancel order

Manage orders

List orders, Add order, Cancel order

Produce stuff

Controlling the factory and monitor its state

1. Manage orders

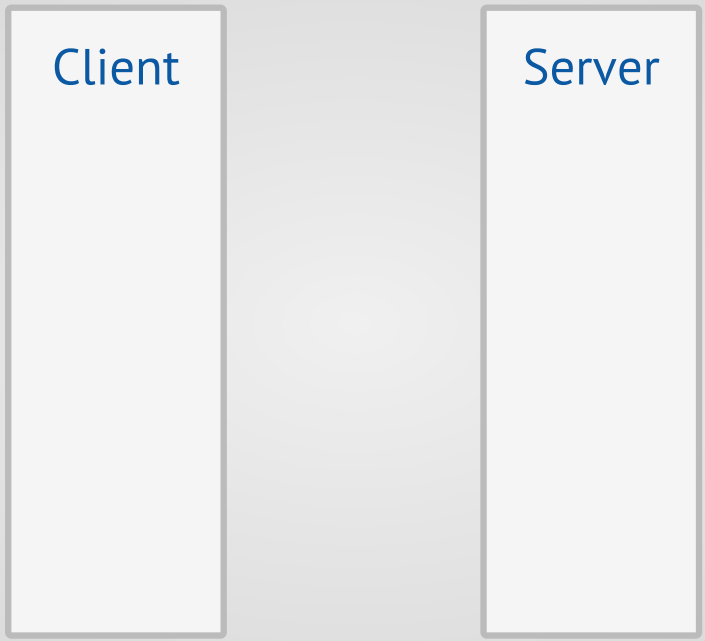
   List orders, Add order, Cancel order

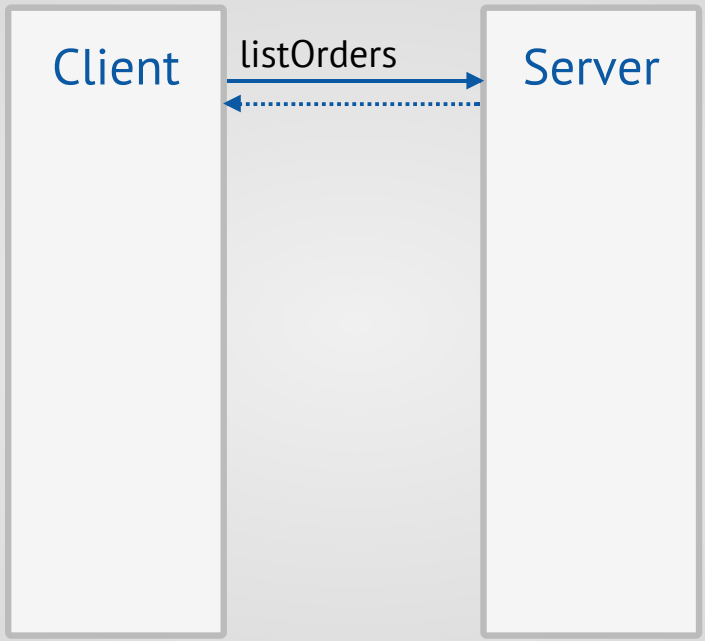
2. Produce stuff

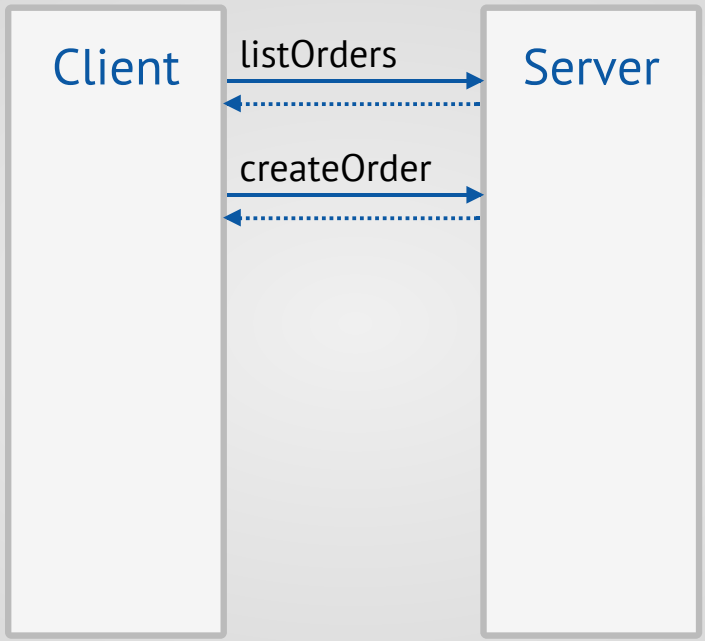
   Controlling the factory and monitor its state

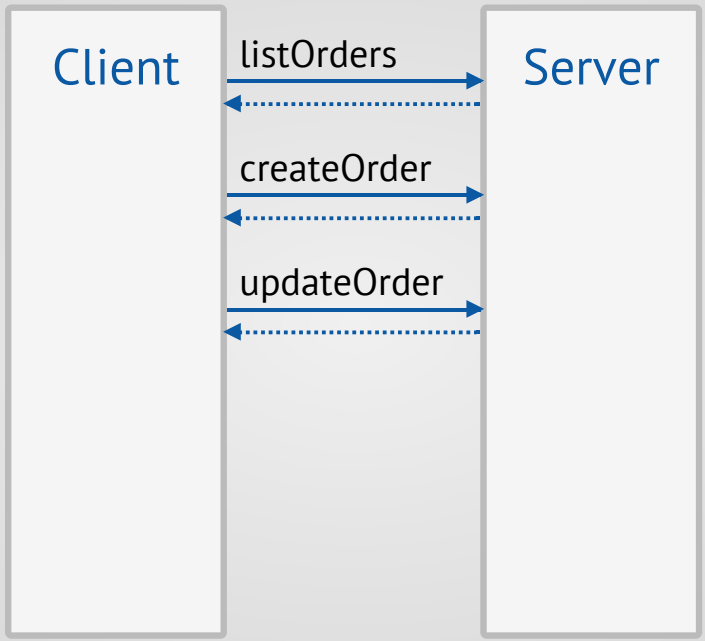
3. Create a UI

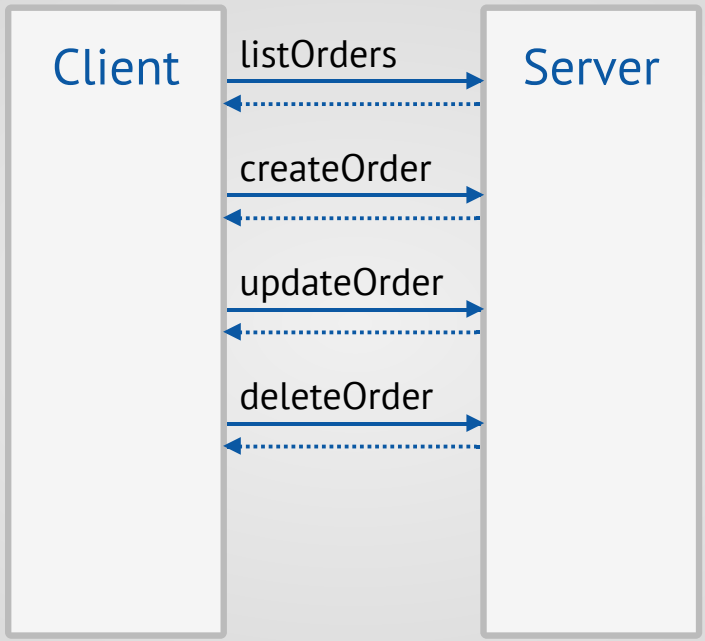
   Shouldn't be that hard


Client
Server


Client
listOrders
Server


Client
listOrders
Server
createOrder


Client
listOrders
Server
createOrder
updateOrder


Client
listOrders
Server
createOrder
updateOrder
deleteOrder

Spring MVC

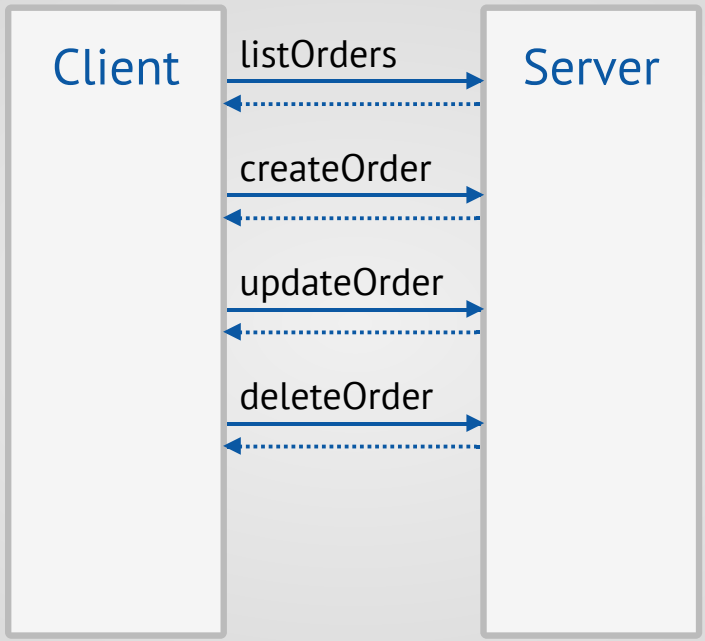
```
@Controller
@RequestMapping(produces = "application/json")
public class OrderController {

    @ResponseBody
    @RequestMapping(value = "/api/order",
                    method = RequestMethod.GET)
    public List<Order> listOrders() { … }

    @ResponseBody
    @RequestMapping(value = "/api/order",
                    method = RequestMethod.POST,
                    consumes = "application/json")
    public Order createOrder(@RequestBody Order order) { … }

    @ResponseBody
    @RequestMapping(value = "/api/order/{id}",
                    method = RequestMethod.PUT,
                    consumes = "application/json")
    public Order updateOrder(@PathVariable Integer id,
```

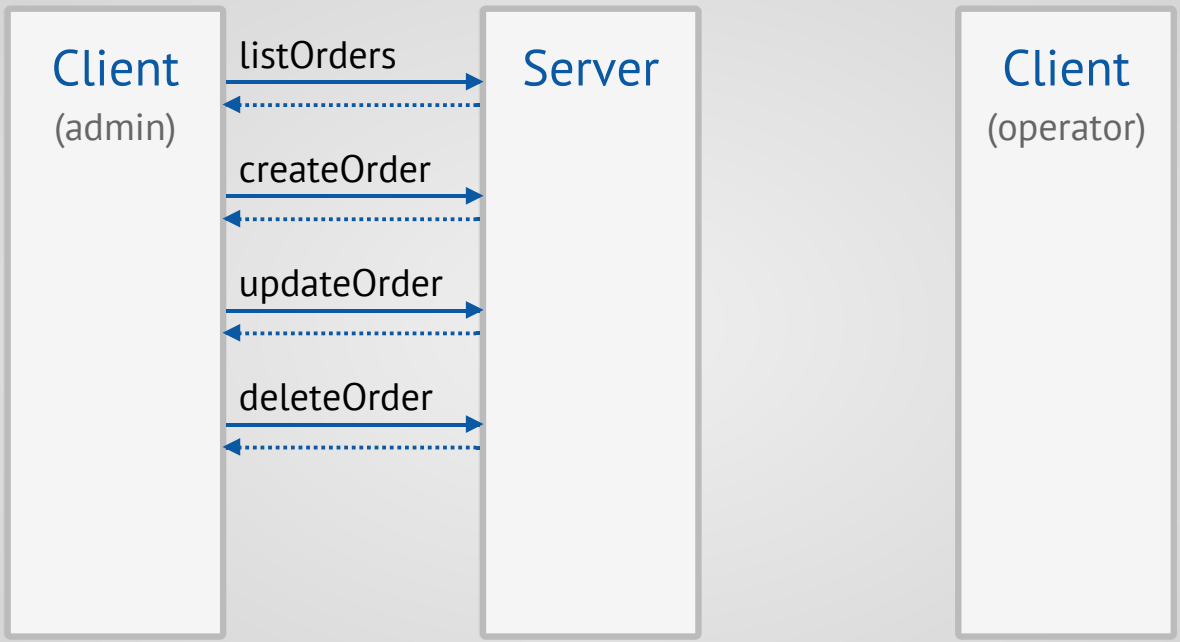
```
                    method = RequestMethod.POST,
                    consumes = "application/json")
    public Order createOrder(@RequestBody Order order) { … }

    @ResponseBody
    @RequestMapping(value = "/api/order/{id}",
                    method = RequestMethod.PUT,
                    consumes = "application/json")
    public Order updateOrder(@PathVariable Integer id,
                             @RequestBody Order order) { … }

    @ResponseBody
    @RequestMapping(value = "/api/order/{id}",
                    method = RequestMethod.DELETE)
    public void deleteOrder(@PathVariable Integer id) { … }
}
```

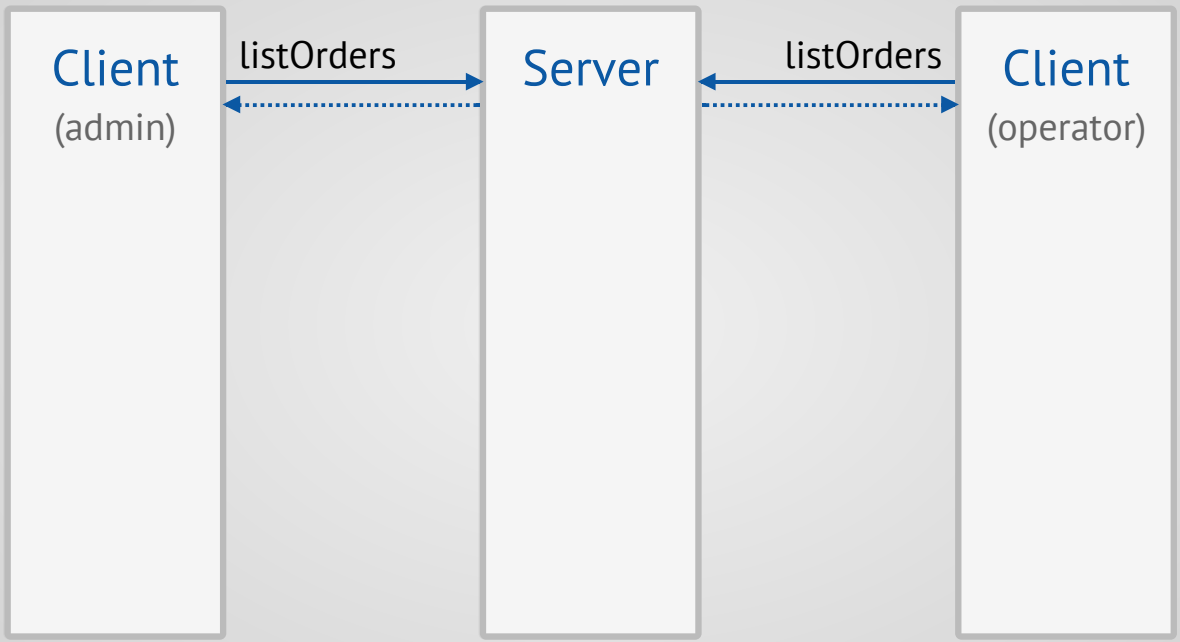
Orders

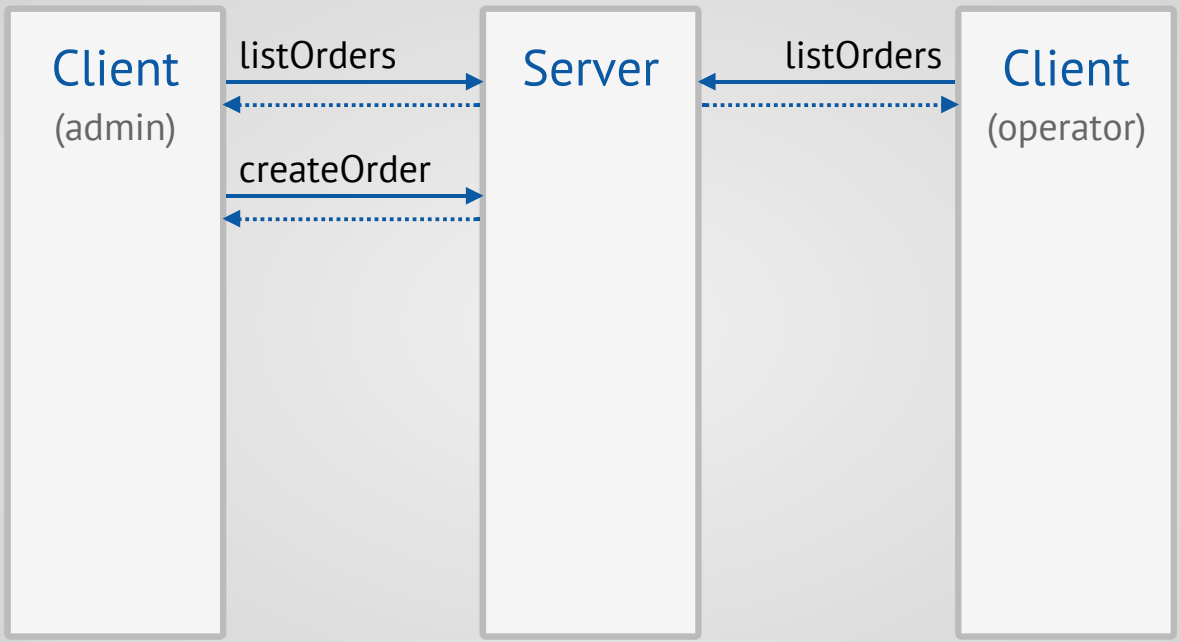
New

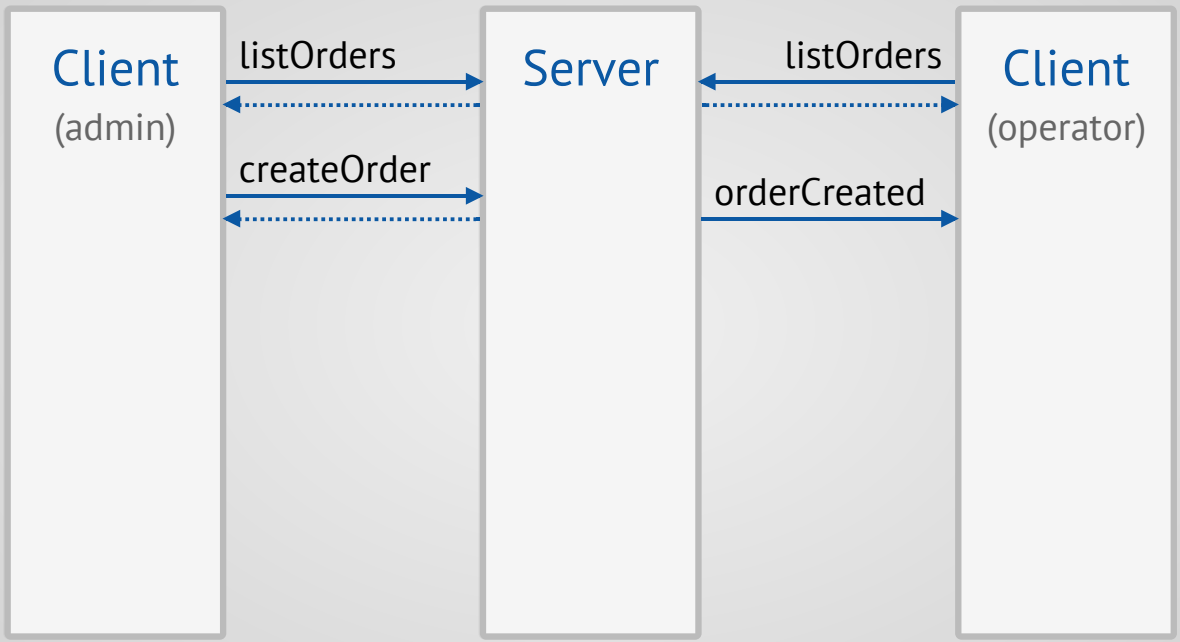
Customer	Quantity	Status
Anna Svensson	100 kg	Received
Jonas Karlsson	150 kg	Received

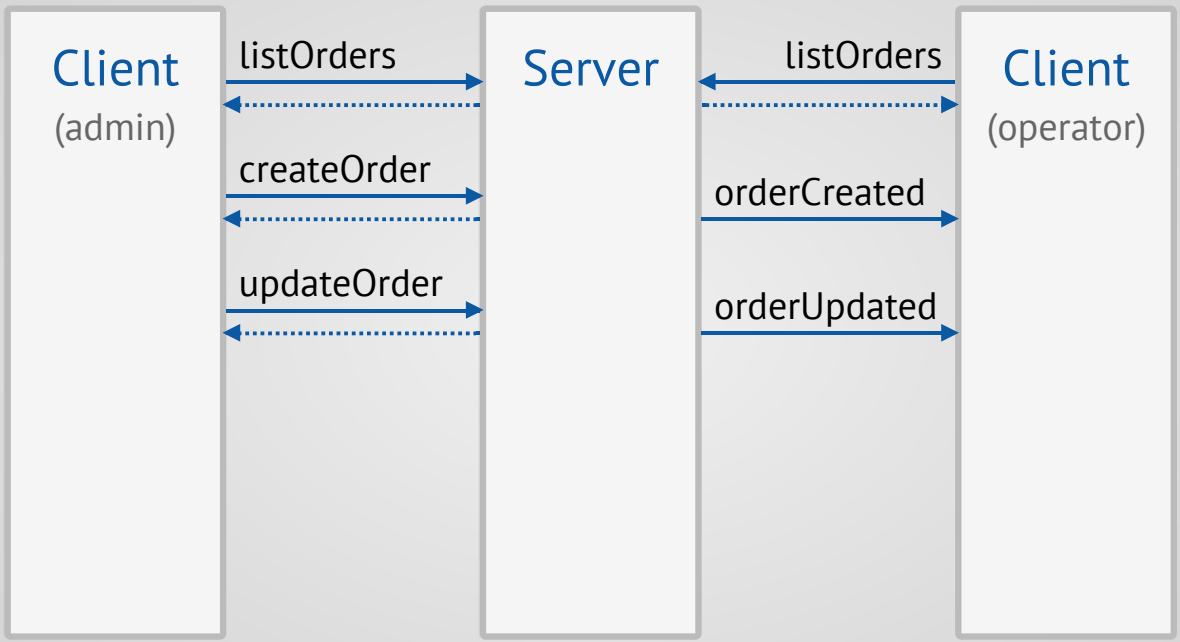

Client
listOrders
Server
createOrder
updateOrder
deleteOrder

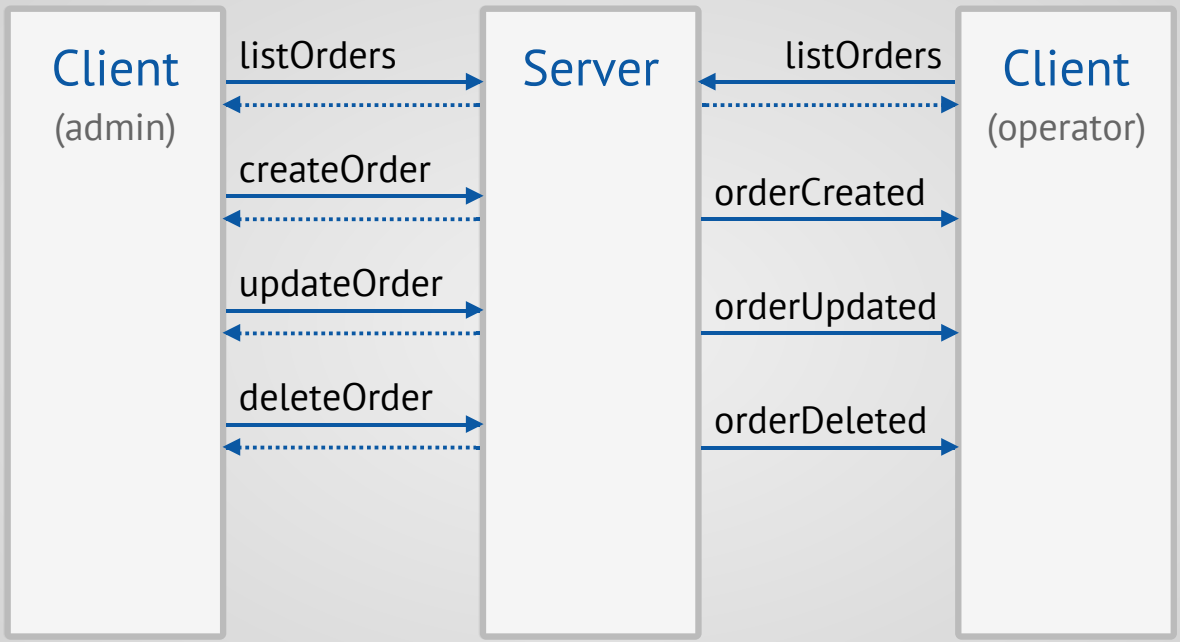

Client
(admin)
listOrders
createOrder
updateOrder
deleteOrder
Server
Client
(operator)

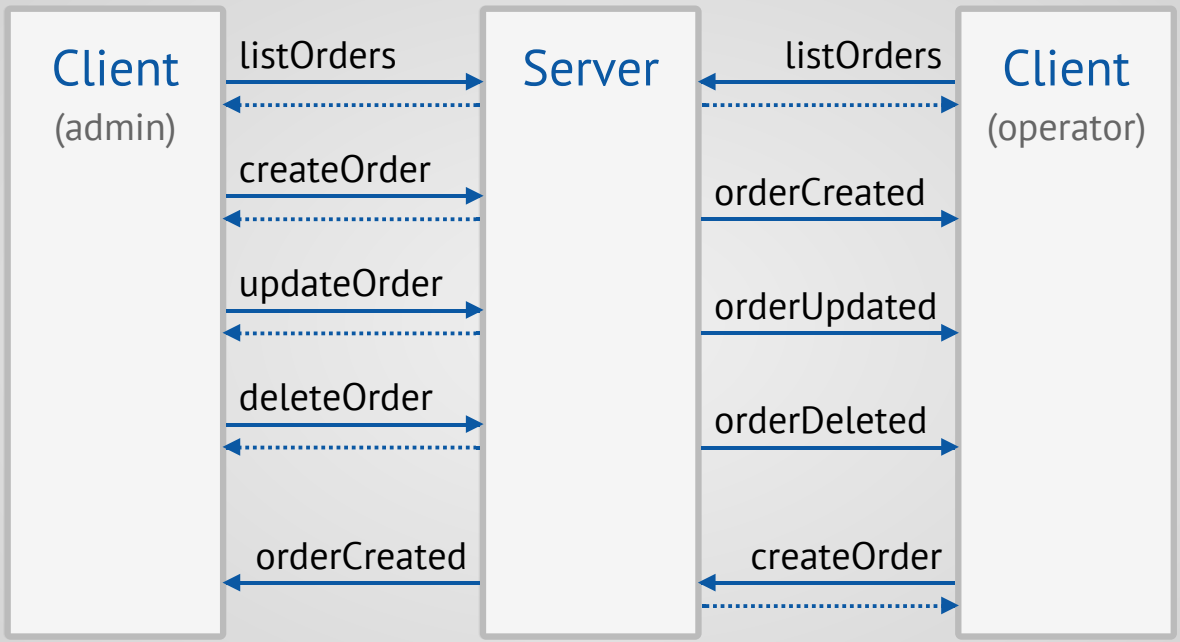

Client
(admin)
listOrders
Server
listOrders
Client
(operator)


Client
(admin)
listOrders
createOrder
Server
listOrders
Client
(operator)


Client
(admin)
listOrders
createOrder
Server
listOrders
orderCreated
Client
(operator)


Client
(admin)
listOrders
createOrder
updateOrder
Server
listOrders
orderCreated
orderUpdated
Client
(operator)


Client
(admin)
listOrders
createOrder
updateOrder
deleteOrder
Server
listOrders
orderCreated
orderUpdated
orderDeleted
Client
(operator)


Client
(admin)
listOrders
createOrder
updateOrder
deleteOrder
orderCreated
Server
listOrders
orderCreated
orderUpdated
orderDeleted
createOrder
Client
(operator)

What is WebSockets?

HTML5

What is WebSockets?

Firewall friendly

Full duplex

What is WebSockets?

2 bytes overhead

WebSocketServlet (Jetty)

```
public class WebSocketHandlerServlet extends WebSocketServlet {

    private WebSocketManager webSocketManager;
    …

    public WebSocket doWebSocketConnect(
            HttpServletRequest request, String protocol) {

        return new WebSocketClient(webSocketManager);
    }
}
```

WebSocketClient (Jetty)

```
public class WebSocketClient implements WebSocket.OnTextMessage {

    private WebSocketManager webSocketManager;
    private Connection connection;

    public WebSocketClient(WebSocketManager webSocketManager) {
        this.webSocketManager = webSocketManager;
    }

    public void onOpen(Connection connection) {
        this.connection = connection;

        webSocketManager.registerClient(this);
    }

    public void onClose(int closeCode, String message) {
        webSocketManager.deregisterClient(this);
    }

    public void onMessage(String message) {
        webSocketManager.onCommand(message, this);
```

```
        webSocketManager.registerClient(this);
    }

    public void onClose(int closeCode, String message) {
        webSocketManager.deregisterClient(this);
    }

    public void onMessage(String message) {
        webSocketManager.onCommand(message, this);
    }

    public void sendMessage(String message) throws IOException {
        if (connection != null) {
            connection.sendMessage(message);
        }
    }

}
```

Protocol

```
{
    protocol : "order",
    command : "created",
    data : {
        id : 1,
        customer : "Anna Svensson",
        quantity : 100,
        status : "Received"
    }
}
```

```
                    method = RequestMethod.GET)
    public List<Order> listOrders() { … }
```

```
    @Autowired
    private WebSocketManager webSocketManager;

    @ResponseBody
    @RequestMapping(value = "/api/order",
                    method = RequestMethod.POST,
                    consumes = "application/json")
    public Order createOrder(@RequestBody Order order) {
        …
        webSocketManager.broadcast("order", "created", order);
    }

    …

}
```

Get the code at **github.com/joakimkemeny**

WebSockets (Spring MVC)

```
@WebSocketController("order")
@Controller
@RequestMapping(produces = "application/json")
public class OrderController {

    @ResponseBody
    @WebSocketMapping("list")
    @RequestMapping(value = "/api/order",
                    method = RequestMethod.GET)
    public List<Order> listOrders() { … }

    @Autowired
    private WebSocketManager webSocketManager;

    @ResponseBody
    @RequestMapping(value = "/api/order",
                    method = RequestMethod.POST,
                    consumes = "application/json")
    public Order createOrder(@RequestBody Order order) {
        …
```

Orders

New

Customer	Quantity	Status
Anna Svensson	100 kg	Received
Jonas Karlsson	150 kg	Received

Backbone.js

MV* library

Backbone.js

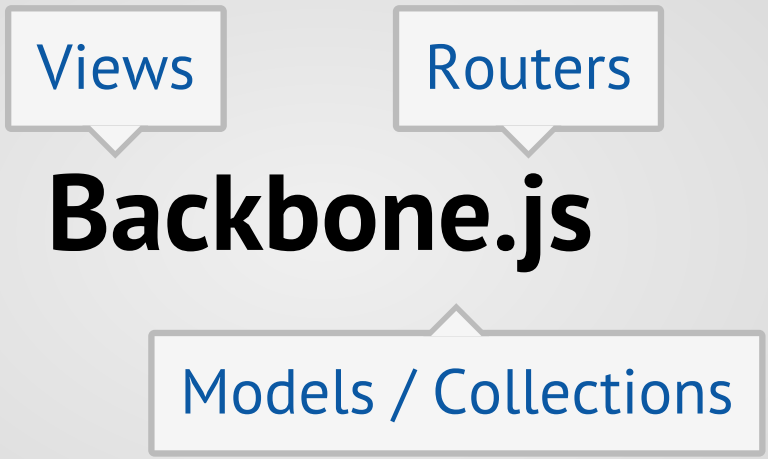
Backbone.js

Models / Collections

Views

Backbone.js

Models / Collections



Read more about Backbone at **backbonejs.org**

Backbone Models / Collections

```
var Order = {};

Order.Model = Backbone.Model.extend({
    urlRoot : "/api/order",

    defaults : {
        customer : null,
        quantity : 100
    }
});


Order.Collection = Backbone.Collection.extend({
    model : Order.Model,
    url : "/api/order"
});
```

```
var orders = new Order.Collection();
orders.fetch();

var order = new Order.Model({
```

```
        quantity : 100
    }
});

Order.Collection = Backbone.Collection.extend({
    model : Order.Model,
    url : "/api/order"
});

var orders = new Order.Collection();
orders.fetch();

var order = new Order.Model({
    customer : "Bo Gren",
    quantity : 100
});
order.save()

order.set("quantity", 200);
order.save();

order.destroy();
```

```
Order.Collection = Backbone.Collection.extend({
    model : Order.Model,
    url : "/api/order"
});
```

```
var orders = new Order.Collection();
orders.fetch();
```

```
var order = new Order.Model({
    customer : "Bo Gren",
    quantity : 100
});
order.save()
```

```
order.set("quantity", 200);
order.save();
```

```
order.destroy();
```

```
var orders = new Order.Collection();
orders.fetch();

var order = new Order.Model({
    customer : "Bo Gren",
    quantity : 100
});
order.save()

order.set("quantity", 200);
order.save();

order.destroy();
```

```
var order = new Order.Model({
    customer : "Bo Gren",
    quantity : 100
});
order.save()

order.set("quantity", 200);
order.save();
```

```
order.destroy();
```

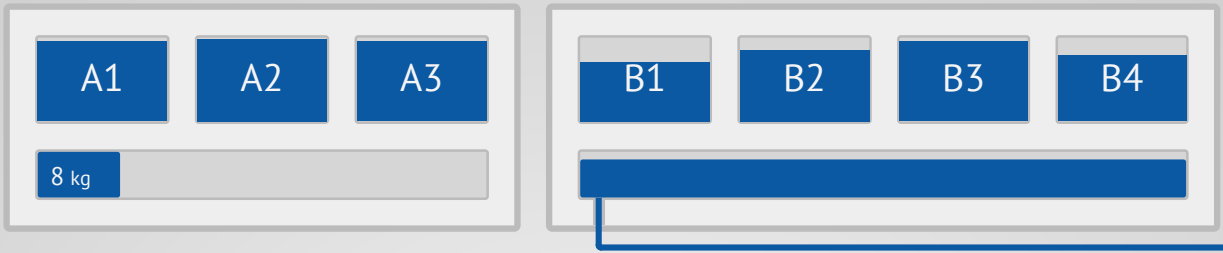
WebSockets (JavaScript)

```
var webSocket = new WebSocket("ws://localhost:8080/ws");

webSocket.onopen = function () { … };

webSocket.onclose = function () { … };

webSocket.onmessage = function (e) {
    var command = JSON.parse(e.data);

    console.log(command.protocol);
    console.log(command.command);

    console.log(command.data);
};

webSocket.send(JSON.stringify(command));
```

WebSockets (Backbone)

```
Order.Collection = Backbone.Collection.extend({
    model : Order.Model,
    url : "/api/order",

    initialize : function () {
        this.listenTo(webSocket, "order:created", this.update);
        this.listenTo(webSocket, "order:updated", this.update);
    },

    update : function (data) { … }
});
```

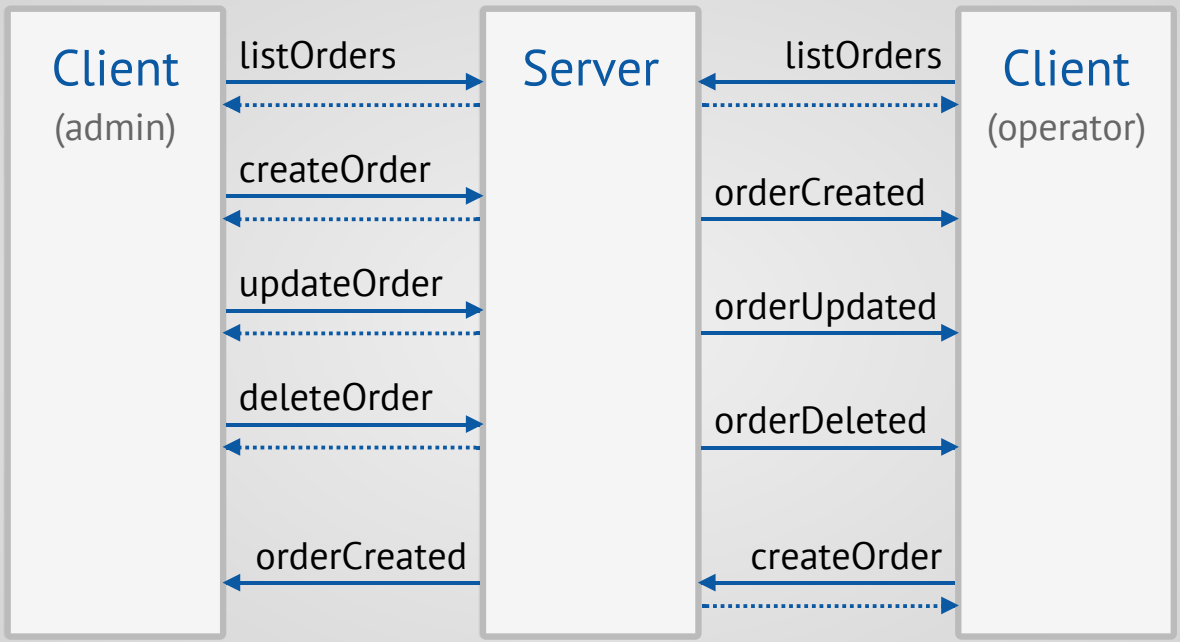
```
    model : Order.Model,
    url : "/api/order",

    initialize : function () {
        this.listenTo(webSocket, "order:created", this.update);
        this.listenTo(webSocket, "order:updated", this.update);
    },

    update : function (data) { … }
});
```

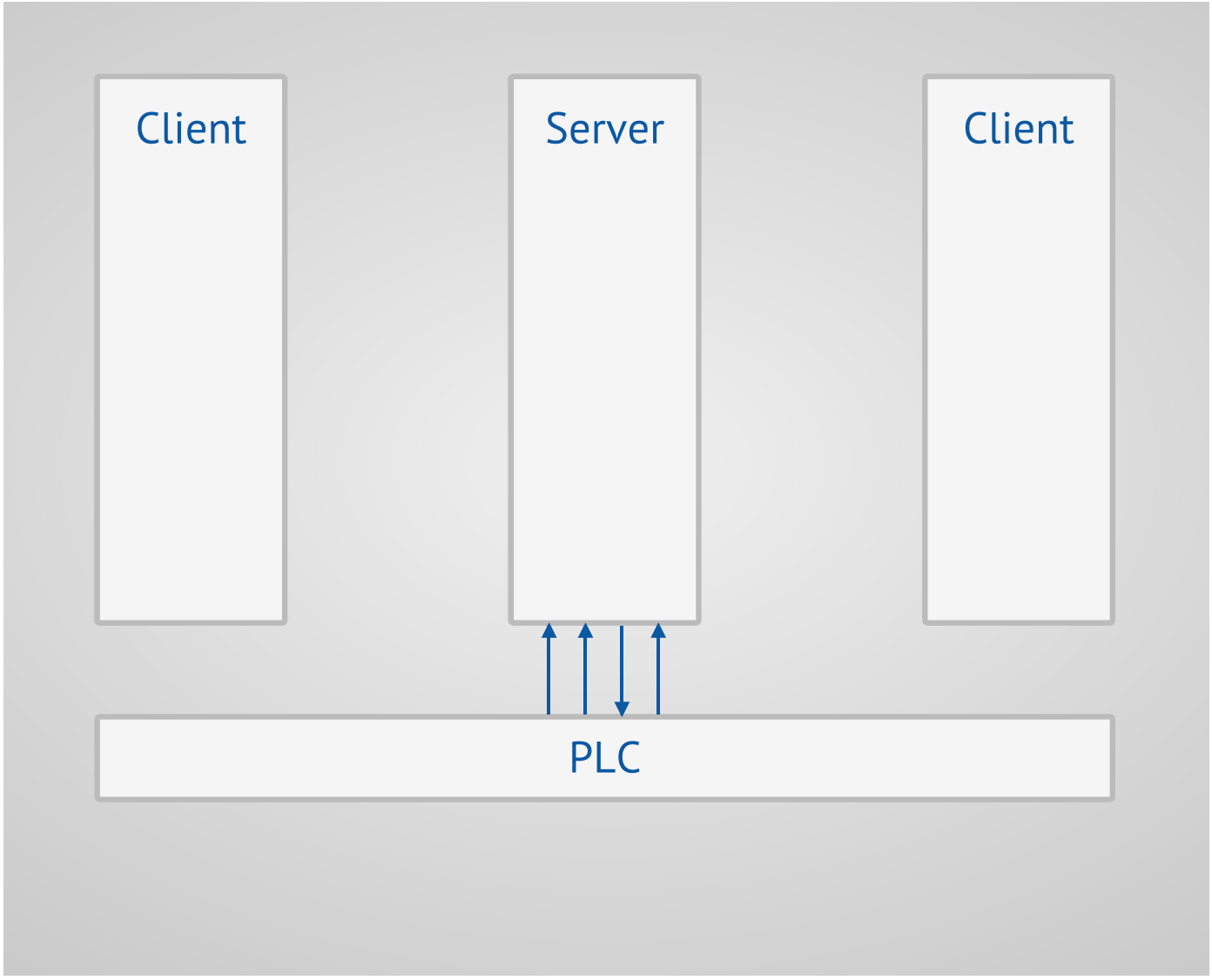


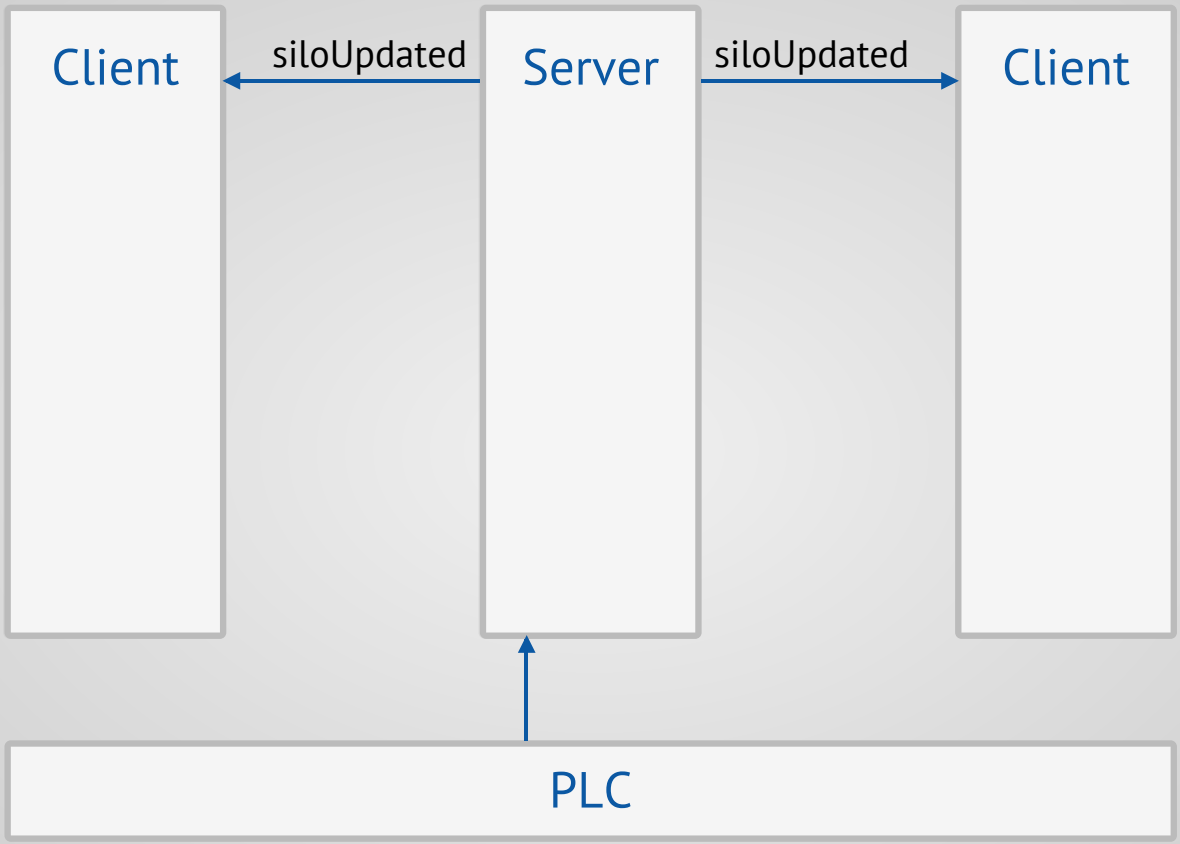
Orders

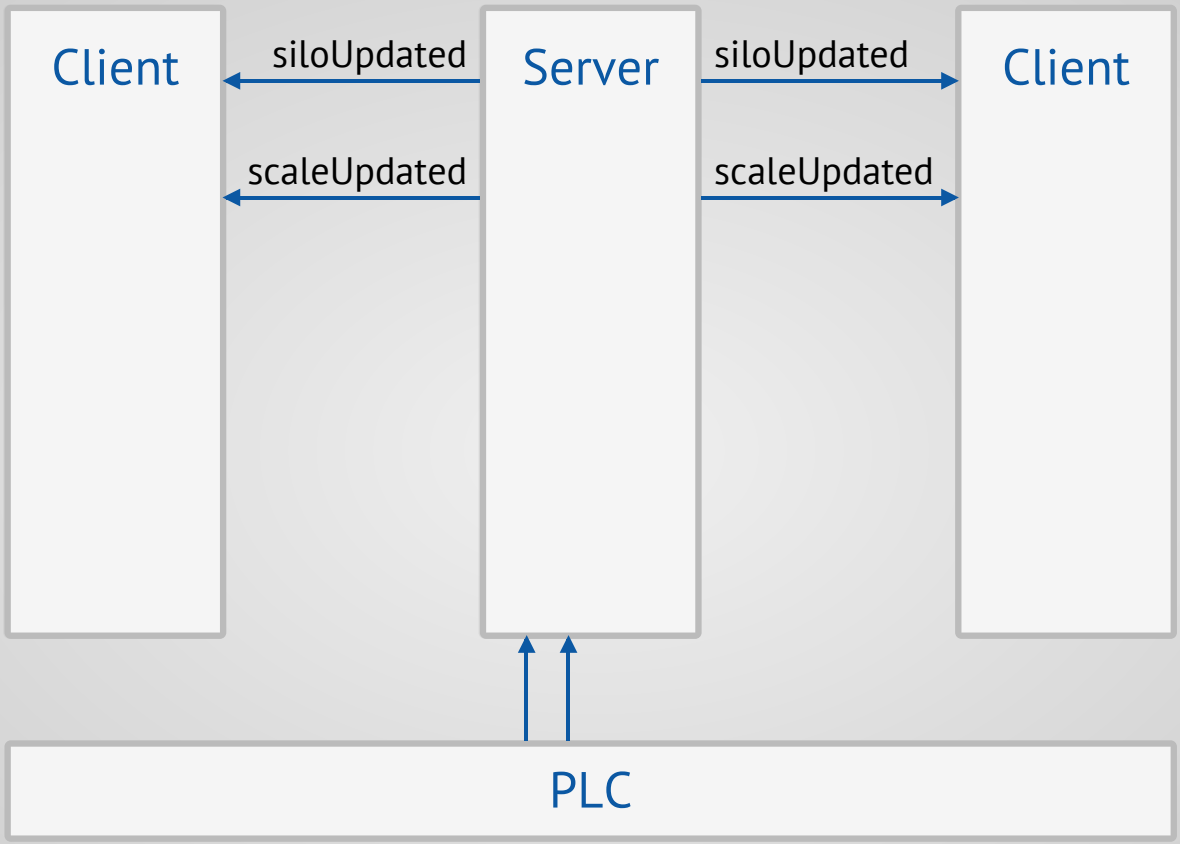
+ New

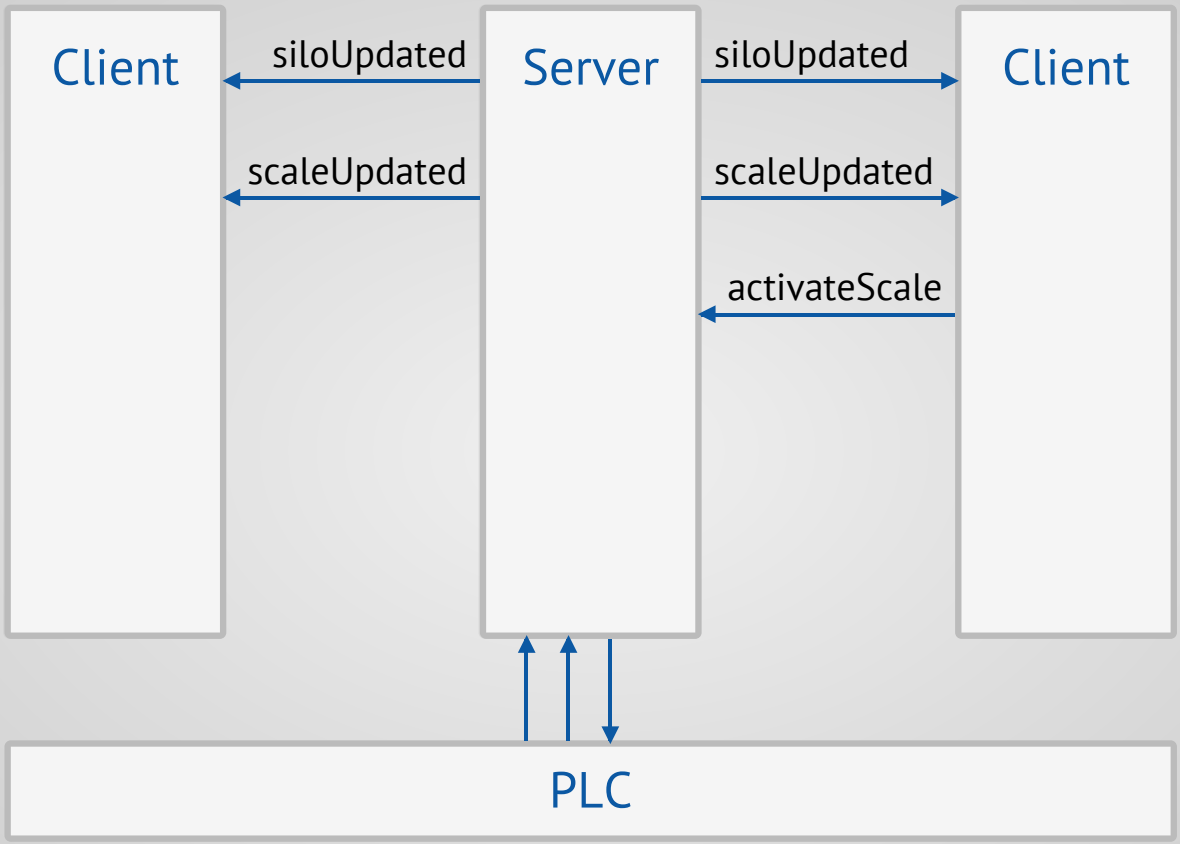
Customer	Quantity	Status
Anna Svensson	100 kg	Producing
Jonas Karlsson	150 kg	Queued
Joakim Kemeny	120 kg	Received


Client
(admin)
listOrders
createOrder
updateOrder
deleteOrder
orderCreated
Server
listOrders
orderCreated
orderUpdated
orderDeleted
createOrder
Client
(operator)


Client
Server
Client
PLC


Client
siloUpdated
Server
siloUpdated
Client
PLC


Client
siloUpdated
Server
siloUpdated
Client
scaleUpdated
scaleUpdated
PLC


Client
siloUpdated
Server
siloUpdated
Client
scaleUpdated
scaleUpdated
activateScale
PLC

Knockout.js

Declarative Bindings

Knockout.js

Declarative Bindings

Knockout.js

Automatic UI Refresh

Read more about Knockout at **knockoutjs.com**

Can work with Backbone...

Knockout.js

Can work with Backbone...

Knockout.js

...using Knockback

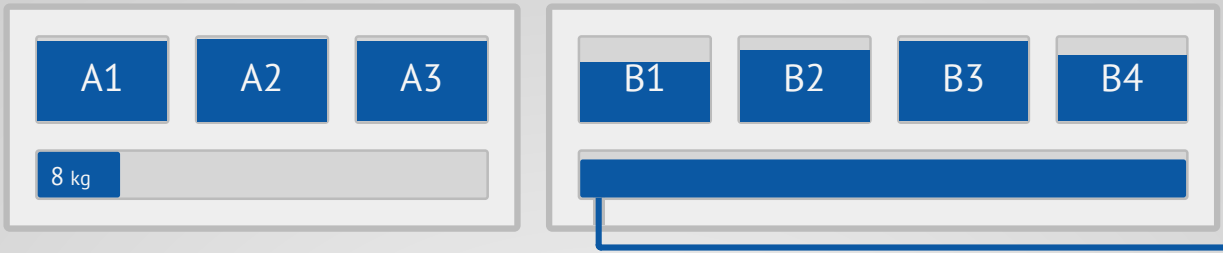
Read more about Knockback at **kmalakoff.github.com/knockback**

Knockout HTML

```
<tbody data-bind="foreach: orders">
<tr>
    <td data-bind="text: customer"></td>
    <td data-bind="text: quantity"></td>
    <td data-bind="text: status"></td>

    <td data-bind="css: {readonly : status() !== 'Received'}">
        <button data-bind="click: $parent.queue"> …
        <button data-bind="click: $parent.edit"> …
        <button data-bind="click: $parent.delete"> …
    </td>
</tr>
</tbody>
```

Knockout HTML

```
<div data-bind="foreach: collection, visibleSlide: !hidden()">
    …
    <div class="scale">
        <div data-bind="visible: !emptying(), style: {width: w}"
             class="filling">
            <span data-bind="text: current"></span>
        </div>
        …
    </div>
</div>
```



Orders

+ New

Customer	Quantity	Status
Anna Svensson	100 kg	Producing
Jonas Karlsson	150 kg	Queued
Joakim Kemeny	120 kg	Received

Conclusions

It takes some hard work

Conclusions

Frameworks need to catch up

JavaScript has its limitations

Conclusions

It works great...

Conclusions

...and it make other UIs feel old

Thank you for coming

WebSockets

Joakim Kemeny
@joakimkemeny
github.com/joakimkemeny