

Clojure-Web-Applikationen für Beginner

BEDCon 2016





Michael Vitz
Consultant @ innoQ

michael.vitz@innoq.com

[@michaelvitz](https://twitter.com/michaelvitz)



Clojure

- › Lisp-Variante auf JVM
- › Funktionale Sprache
- › Dynamisch typisiert
- › Immutable Datenstrukturen
- › komplettes mächtiges Macrosystem

Datenstrukturen

3 3.14 3/2

“Hello World” \a #“Ch.*se”

foo :first

(“Hello”, :first)

{ :name “Michael”,
:company “innoQ” }

[3 4 3]

#{3 4 3}

Syntax

“You’ve just seen it”

- Rich Hickey

Funktionen

(+ 1 2)

› 3

(:city {:name “innoQ”
 :city “Monheim”})

› “Monheim”

(map inc [1 2 3])

› (2 3 4)

Funktionen

```
(fn [x y] (+ x y))
```

```
(def add  
  (fn [x y] (+ x y)))
```

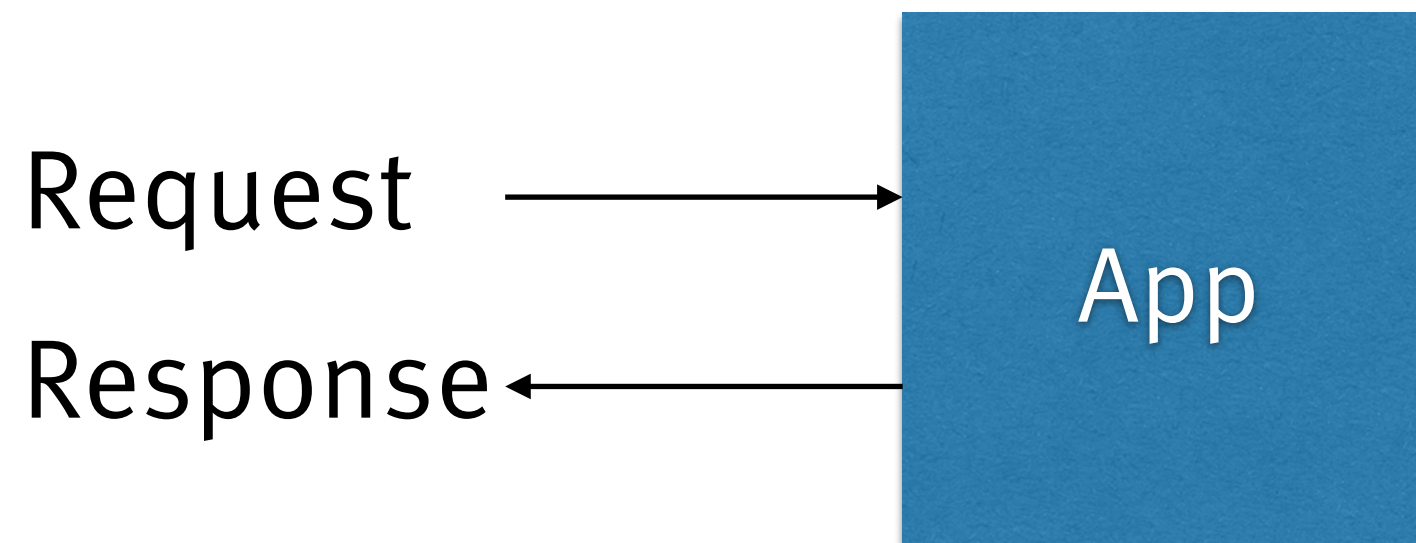
```
(defn add  
  [x y]  
  (+ x y))
```


Links

- › <http://clojure.org>
- › blog.cognitect.com/blog/2016/1/28/state-of-clojure-2015-survey-results
- › <https://juxt.pro/radar.html>
- › <http://www.clojure-toolbox.com>

Web-Applikationen

Web-Applikation



Web-Applikation

$\text{App}(\text{Request}) = \text{Response}$

Web-Applikation

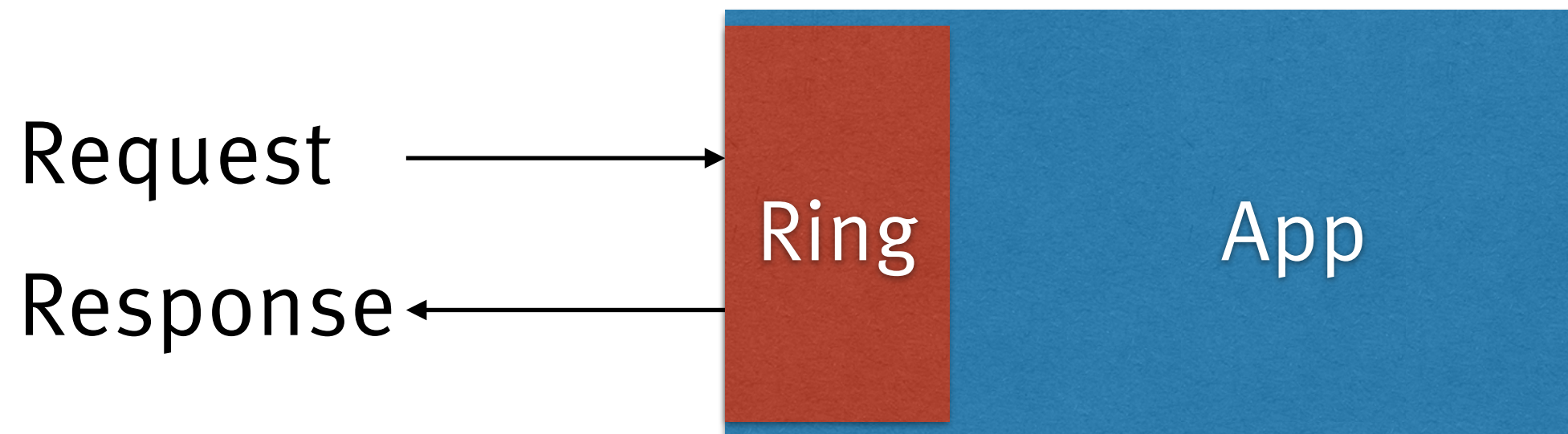
```
(defn greet-controller  
  [request]  
  {:status 200  
   :headers {"Content-Type" "text/plain"}  
   :body "Hello, world!"})
```

Ring

<https://github.com/ring-clojure/ring>

 @michaelvitz

Ring



Ring: Request

```
{:uri “/greet”  
  :query-string “name=Michael”  
  :request-method :get  
  :headers { ... }  
  ...}
```


Ring: Response

```
{:status 200  
 :headers { ... }  
 :body “Hello, Michael!”}
```

Ring: Handler

```
(defn greet-controller  
  [request]  
  {:status 200  
   :headers {"Content-Type" "text/plain"}  
   :body "Hello, world!"})
```

Ring: Adapter

- › Jetty
- › Servlet
- › http-kit
- › Tomcat
- › ...

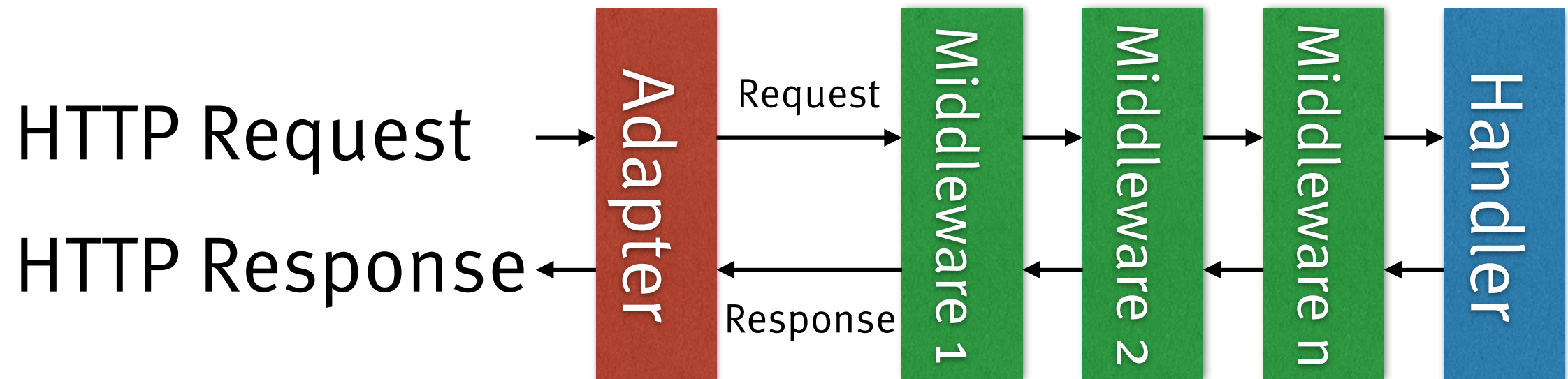
Ring: Middleware

```
(defn wrap-logging  
  [handler]  
  (fn [request]  
    (print request)  
    (let [response (handler request)]  
      (print response)  
      response))))
```

Ring: Middleware

- › <https://github.com/ring-clojure/ring-defaults>
- › <https://github.com/ring-clojure/ring-json>

Ring

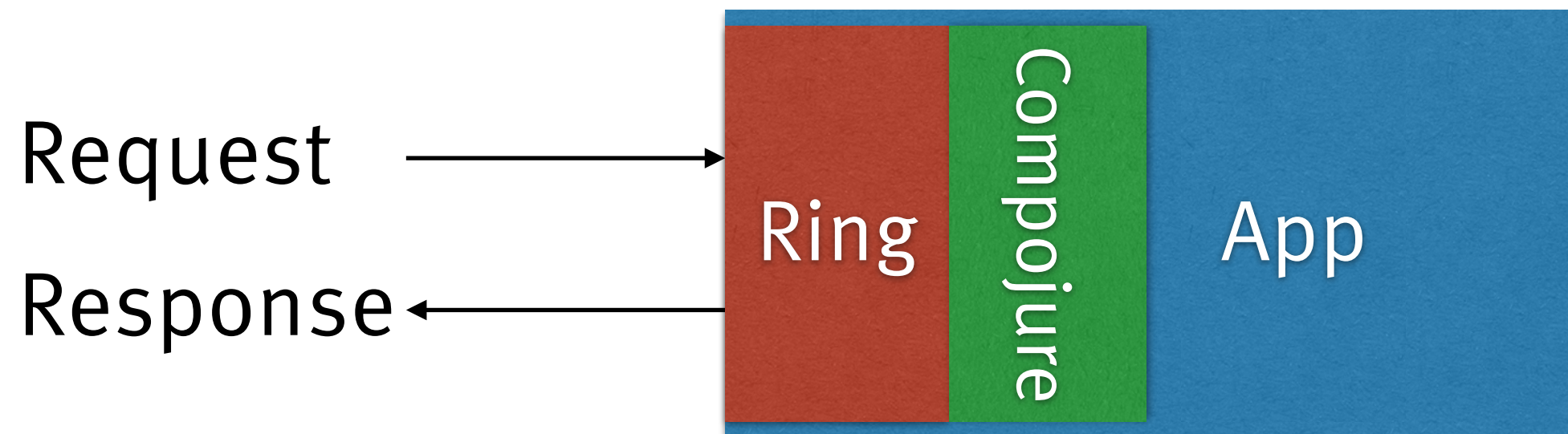


Compojure

<https://github.com/weavejester/compojure>

 @michaelvitz

Compojure



Compojure: Handler

```
(def handler
```

```
  (GET “/hello” [] “Hello, world!"))
```

```
(handler {:request-method :get, :uri “/hello”})
```

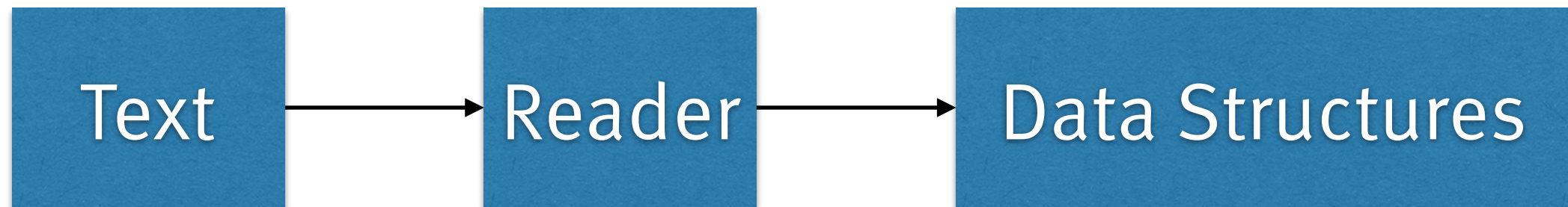
```
> {:body “Hello, world!”}
```

```
(handler {:request-method :post, :uri “/hello”})
```

```
> nil
```

Exkurs: Macros

Clojure: Macros



“(case 3
1 “one”
2 “two”
“more”)”

(case 3
1 “one”
2 “two”
“more”)

Clojure: Macros

Data Structures

Evaluator

Data Structures

```
(case 3  
  1 "one"  
  2 "two"  
  "more")
```

```
(if (= 3 1)  
  "one"  
  (if (= 3 2)  
    "two"  
    "more"))
```

Zurück zu Compojure...

Compojure: Macro

```
(GET “/hello” [] “Hello, world!”)
```

```
(fn [req]  
  (if (and (= (:uri req) “/hello”)  
           (= (:request-method req) :get))  
    { :body “Hello, world!” })))
```

Compojure: Path

```
(GET “/hello/:name” [name]  
  (str “Hello, ” name “!”))
```


Compojure: Routes

```
(def my-routes  
  (routes  
    (GET "/hello" [] "Hello!")  
    (GET "/bye" [] "Bye!")))
```

```
(defroutes my-routes  
  (GET "/hello" [] "Hello!")  
  (GET "/bye" [] "Bye!"))
```


Compojure: Alternativen

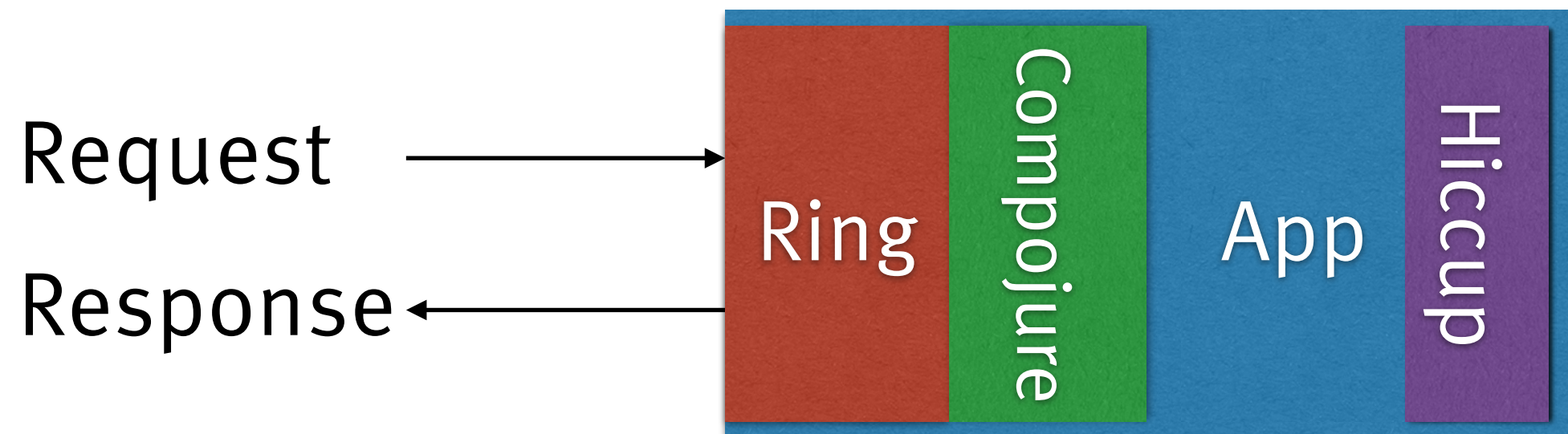
- › <https://github.com/juxt/bidi>
- › <http://clojure-liberator.github.io/liberator/>

Hiccup

<https://github.com/weavejester/hiccup>

 @michaelvitz

Hiccup



Hiccup: Basics

`[:div {:id "foo"}]`

`[:span "bar"]]`

`<div id="foo">`

`<span>bar</span>`

`</div>`

Hiccup: Shortcuts

`[:div#foo`

`[:span "bar"]]`

`<div id="foo">`

`<span>bar</span>`

`</div>`

Hiccup: Links

(link-to “www.innoq.com” “innoQ”)

[:a { :href “www.innoq.com” } “innoQ”]

innoQ

Hiccup: Formulare

```
(form-to [:post “/login”]  
  (text-field “Username”)  
  (password-field “Password”)  
  (submit-button “Login”))
```


Hiccup: Alternativen

- › <https://github.com/danlarkin/clabango>
- › <https://github.com/plakat/clj-thymeleaf>
- › <https://github.com/cgrand/enlive>
- › <https://github.com/yogthos/Selmer>

Fazit

Fazit

- › Funktionale Sprache ideal für Web-App
- › Verbreitung nimmt zu
- › Libraries anstelle von Frameworks
- › Community gesund und hilfsbereit

Interessante Libraries

- › <https://github.com/technomancy/leiningen>
- › <https://github.com/weavejester/environ>
- › <https://github.com/clojurewerkz/route-one>
- › <https://github.com/krisajenkins/yesql>
- › <https://github.com/seancorfield/clj-time>

Frameworks

- › <https://github.com/weavejester/duct>
- › <http://www.luminusweb.net>
- › <https://github.com/otto-de/tesla-microservice>
- › <https://github.com/metosin/compojure-api>

Danke!

Fragen?

Kommentare?

Michael Vitz |  @michaelvitz

michael.vitz@innoq.com

<https://www.innoq.com/de/talks/2016/09/clojure-web-applikationen>



innoQ Deutschland GmbH

Krischerstr. 100
40789 Monheim am Rhein
Germany
Phone: +49 2173 3366-0

Ohlauer Straße 43
10999 Berlin
Germany

Ludwigstraße 180 E
D-63067 Offenbach
Germany

Kreuzstr. 16
D-80331 München
Germany